RADC-TR-79-266
Final Technical Report
October 1979

ROME AIR DEVELOPMENT CENTER

COMPUTER ROUTINES FOR USE IN SECOND ORDER VOLTERRA IDENTIFICATION OF EMI

University of South Florida

V. K. Jain
J. S. Osman

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-266 has been reviewed and is approved for publicatior

APPROVED: John F. Spina

JOHN F. SPINA
Project Engineer

APPROVED: David C. Luke

DAVID C. LUKE, Lt Col, USAF
Chief, Reliability & Compatibility Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBCT), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. ~~REPORT NUMBER~~ (18) RADC-TR-79-266

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. ~~TITLE (and Subtitle)~~ (6) COMPUTER ROUTINES FOR USE IN SECOND-ORDER VOLTERRA IDENTIFICATION OF EMI.

5. ~~TYPE OF REPORT & PERIOD COVERED~~ (9) Final Technical Report, 1 Oct 77 — 30 Sep 78

6. ~~PERFORMING ORG. REPORT NUMBER~~ SS-18VRA

7. AUTHOR(s)
(10) V. K. Jain
J. S. Osman

8. ~~CONTRACT OR GRANT NUMBER(s)~~ (15) F30602-75-C-0118

9. PERFORMING ORGANIZATION NAME AND ADDRESS
University of South Florida
Department of Electrical Engineering
Tampa FL 33620 ✓

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
62702F
(16) 454002P1 (17) Ø21

11. CONTROLLING OFFICE NAME AND ADDRESS
Rome Air Development Center (RBCT)
Griffiss AFB NY 13441

12. ~~REPORT DATE~~ (11) Nov[redacted]79

13. NUMBER OF PAGES
91

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
Same
(12) 89

15. SECURITY CLASS (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION DOWNGRADING SCHEDULE
N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John F. Spina (RBCT)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pencil-of-functions method
System identification
Parameter estimates
Integrators
Volterra system
Transfer functions
Gram matrix
Residues of quadratic transfer function
Ill-conditioned matrix
Adaptive scaling
Perturbation theory
Iterative correction
Matrix inversion

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Computer routines are developed for application in Volterra modeling of weakly nonlinear systems. The FORTRAN program "IGRAM" identifies the parameters of a linear black-box model as given by the pencil-of-functions method. The FORTRAN program "HPMINV" yields high accuracy inversion of residue matrices that arise in Volterra system identification.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

4Ø8631

PREFACE

This effort was conducted by University of South Florida under the sponsorship of the Rome Air Development Center Post-Doctoral Program for Rome Air Development Center. Mr. John F. Spina RADC/RBCT was the task project engineer and provided overall technical directicn and guidance.

The RADC Post-Doctoral Program is a cooperative venture between RADC and some sixty-five universities eligible to participate in the program. Syracuse University (Department of Electrical Engineering), Purdue University (School of Electrical Engineering), Georgia Institute of Technology (School of Electrical Engineering), and State University of New York at Buffalo (Department of Electrical Engineering) act as prime contractor schools with other schools participating via sub-contracts with prime schools. The U.S. Air Force Academy (Department of Electrical Engineering), Air Force Institute of Technology (Department of Electrical Engineering), and the Naval Post Graduate School (Department of Electrical Engineering) also participate in the program.

The Post-Doctoral Program provides an opportunity for faculty at participating universities to spend up to one year full time on exploratory development and problem-solving efforts with the post-doctorals splitting their time between the customer location and their educational institutions. The program is totally customer-funded with current projects being undertaken for Rome Air Development Center (RADC), Space and Missile Systems Organization (SAMSO), Aeronautical System Division (ASD), Electronics Systems Division (ESD), Air Force Avionics Laboratory (AFAL), Foreign Technology Division (FTD), Air Force Weapons Laboratory (AFWL), Armament Development and Test Center (ADTC), Air Force Communications Service (AFCS), Aerospace Defense Command (ADC), HW USAF, Defense Communications Agency (DCA), Navy, Army, Aerospace Medical Division (AMD), and Federal Aviation Administration (FAA).

Further information about the RADC-Doctoral Program can be obtained from Mr. Jacob Scherer, RADC/RBC, Griffiss AFB, NY, 13441, telephone Autovon 587-2543, Commercial (315) 330-2543.

ACKNOWLEDGEMENTS

The authors are deeply indebted to Mr. John. F. Spina and Mr. D. J. Kenneally for their assistance and useful criticism throughout the duration of this project. They also wish to express their gratitude to Dr. Donald D. Weiner for the many stimulating and helpful discussions on different aspects of this work. Appreciation is also extended to Mr. Jacob Scherer of the Post-Doc Program for his helpfulness.

The authors also acknowledge the assistance of Owen L. Godwin, Jr. in certain parts of this report.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter II

Chapter III

Appendix C

COMPUTER ROUTINES FOR USE IN SECOND-ORDER VOLTERRA MODELING OF EMI

I. INTRODUCTION

Research described here consists of the development of certain computer routines to aid Volterra modeling of weakly nonlinear systems [1], [2]. Volterra series representation, a dynamic generalization of the familiar power series, is ideal for representing devices and systems with frequency-dependent mild nonlinearities as in the case of a transistor. The technique has been applied to the analysis of communication receiver response to radio frequency interference[3]. Other applications include intermodulation distortion analysis of transistor feedback amplifiers [4], nonlinear characterization of IMPATT diodes and microwave amplifiers [5], and analysis of channels with soft limiter.

Rome Air Development Center has in the recent past supported several efforts to put this analytical tool to use in the electromagnetic interference and compatibility field. (In practical terms one of the major outcomes of the efforts is the IAP program, a computer program for the prediction of Intra-System Electromagnetic Compatibility). A current direction of interest is the estimation of Volterra kernels of a system from its experimentally observed input-output responses. Interest in this black-box approach arises for several reasons, the most salient being cost effectiveness in testing, and simplicity of resulting models for complex on-board communication systems.

Weiner and Ewen [1], [2] have provided an approach to finding the parameters of the kernels, specifically the poles and residues of the multivariable transfer functions $H_m(s_1,....,s_m)$. The poles and residues of the linear TF, $H_1(s)$, are determined using Jain's method [6], [7](pencil-of-functions identification method). Then, for somewhat larger amplitudes, where the quadratic TF $H_2(s_1,s_2)$ has non-negligible influence, the contribution $y_2(t)$ is determined by subtracting $y_1(t)$, the predicted response of $H_1(s)$, from $y(t)$. The poles of the quadratic TF are known in terms of the poles of the linear TF, so that only the residues of $H_2(s_1, s_2)$ need be -- and in principle, can be -- determined. A similar procedure is adopted for determining the parameters of the cubic kernel, and so on.

The computer programs presented in this report are:

IGRAM	Program for black-box identification of a linear transfer function using pencil-of-functions method.
HPMINV	High precision matrix inversion routine for use in determination of the residues of the quadratic Volterra TF. Perturbation theory and iterative corrections are used to enable accurate inversion even for wideband system matrices.

II. PENCIL OF FUNCTIONS METHOD FOR IDENTIFICATION OF NETWORK TRANSFER FUNCTIONS

Determining the model of a network from its observed input-output responses represents the inverse of the analysis problem. Interest in this arises from the frequent need for a relatively simple mathematical description of the system so that behavior for other anticipated inputs may be predicted up to acceptable accuracies. Like the analysis problem, there are several approaches available in the literature for the inverse, or, as it is often called, the "identification" problem . To name a few, (a) Prony's method [8], (b) gradient methods, such as Newton [9] and quasi-linearization [10],(c) least-squares and generalized least-squares methods [11],[12],(d) maximum-likelihood methods [13],[14],[15].

All of the methods stated above possess certain advantages and, as may be expected, certain disadvantages peculiar to each particular method. Stated very broadly, sensitivity to noise, slow convergence to the solution, and excessive computational complexity are some of the possible disadvantages. The objective of this section is to describe in a semi-rigorous way the pencil-of-functions identification method [7]. Further, in this section the method is extended to the case where general first order filters, $\mu_i(s) = (b_i s + c_i)/(s+a_i)$ are used in the processing system instead of ideal integrators (note that the ideal integrator is a special case of this filter; set $b_i = a_i = 0$). A high accuracy FORTRAN program, developed for the case of ideal-integrator processing units, is also presented.

The method offers the advantages of mathematical simplicity, closed-form solution to the problem, which is optimal in the generalized least-squares sense and suboptimal in the strict least-squares sense, and robustness to noise. The disadvantage of the method is that the variances of additive noise, when present, must be determined in a separate experiment in order that unbiased parameter estimates may be computed.

2.1 THEORY

The problem of identifying the transfer function of a network from its input-output responses can be formulated in discrete-time domain as follows. Given the pair x(k), y(k) (or noise-corrupted versions of these; call them u(k), v(k)) find the transfer function H(z) that produces a response matching y(k) (or v(k)) when excited by x(k) (or u(k)). More specifically, this involves determination of the parameters a_i, b_i in

$$Y(z) = H(z)\,X(z)$$

$$= \left(\frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}}\right) X(z) \tag{1}$$

from experimental input-output data. Note (1) can be written in time domain as

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{i=0}^{n} b_i\, x(k-i),\ y(k) = 0 \text{ for } k \leq 0 \tag{2}$$

A. Measurement Signals

Before proceeding with the solution of the problem via pencil-of-function method, let us note that (2) can be written as

$$[y(k)\ y(k-1)\ .\ .\ .\ y(k-n)\ -x(k)\ -x(k-1)\ .\ .\ .\ -x(k-n)]\ \underline{\theta} = 0 \tag{3}$$

or, more concisely,

$$\underline{f}(k)^T\ \underline{\theta} = 0 \tag{4}$$

where the 2n + 2 dimensional vector $\underline{f}(k)$ has the obvious definition and the vector $\underline{\theta}$, also 2n + 2 dimensional, is given as

$$\underline{\theta} = \begin{bmatrix} \underline{a} \\ \\ \underline{b} \end{bmatrix} \tag{5}$$

Equation (4) represents a geometrical constraint upon the vectors $\underline{f}(k)$, namely that they are all orthogonal to $\underline{\theta}$.

To cast the problem of identifying $\underline{\theta}$ into a generalized least-squares problem consider the measurement system of Fig. 1. The matter of choosing the first order filters which make up the measurement units $M_i(z)$ will not be considered here. It will suffice to say they must be chosen so that the cutoff frequencies of $M_i(z)$ span the frequency range of interest (i.e. they are spread in the pass-band of the network under test). Regardless of the choice of the measurement units the following useful observation arises.

Let

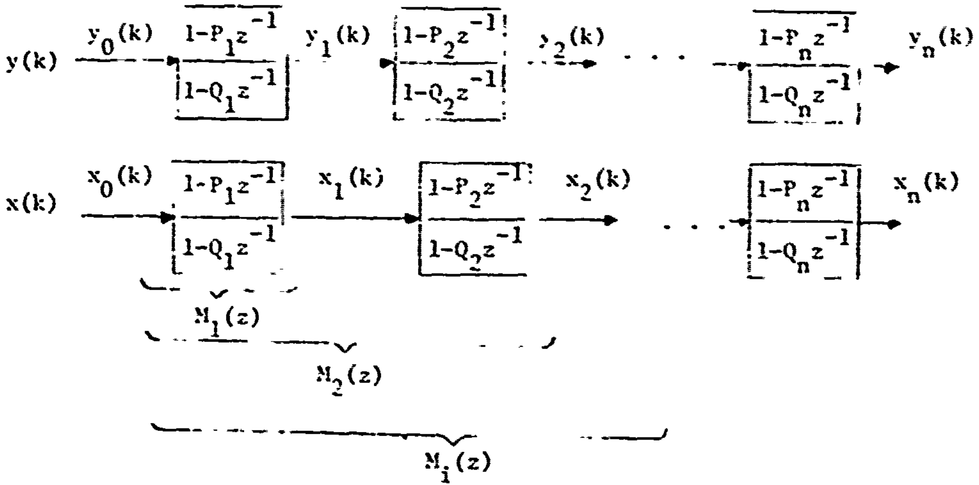
$$\underline{\phi}(k) = [y_o(k)\ y_1(k)\ \ldots\ y_n(k)\ -x_o(k)\ -x_1(k)\ \ldots\ -x_n(k)]^T$$

and define the matrix

$$C = \begin{bmatrix} c_{00} & c_{01} & \cdots & c_{0n} \\ c_{10} & c_{11} & & c_{1n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ c_{n0} & c_{n1} & & c_{nn} \end{bmatrix} \tag{6a}$$

where c_{ji} are the coefficients of the polynomial

$$\sum_{i=0}^{n} c_{ji} z^{-1} = \prod_{\ell=1}^{i} (1-P_\ell z^{-1}) \prod_{\ell=i+1}^{n} (1-Q_\ell z^{-1}) \tag{6b}$$



Measurement vector $[y_0(k)\ \ldots.\ y_n(k)\ \ -x_0(k)\ \ldots.\ -x_n(k)]^T = \underline{\phi}(k)$

First order filters $M_i(z) = (1-P_iz^{-1})/(1-Q_iz^{-1})$

Fig. 1. Measurement system

Also define the vector $\underline{\gamma}$ henceforth called the synthetic parameter vector, as

$$\underline{\gamma} = \begin{bmatrix} \underline{\alpha} \\ \underline{\beta} \end{bmatrix} \tag{7a}$$

$$= \begin{bmatrix} C^{-1}\underline{a} \\ C^{-1}\underline{b} \end{bmatrix} \tag{7b}$$

$$= C^{-1}\theta \tag{7c}$$

Then, it can be shown that the measurement vectors $\underline{\phi}_k$ are orthogonal to the synthetic parameter vector. The proof will be given in subsection 2.1C.

B. Solution

The problem of determining the synthetic parameter vector γ can be shown to reduce to finding the cofactors of a gram matrix. Specifically, it can be shown by use of the pencil-of-functions theorem [6] that the vector γ can be obtained in the following way:

1. Form the Gram matrix [16]

$$G_N = \sum_{k=0}^{K-1} \underline{\phi}(k)\underline{\phi}^T(k) ,$$

where $N = 2n + 2$

2. Find the diagonal cofactors of G_N; call them Λ_{ii}
3. Then

$$\underline{\gamma}^T = \frac{1}{\sqrt{\Lambda_{11}}}\left[\sqrt{\Lambda_{11}} , \cdots , \sqrt{\Lambda_{NN}}\right]$$

Finally, using the transformation (7c) we have the desired parameter vector $\underline{\theta} = C\underline{\gamma}$

C. Proof of Measurement Filter Theorem

The relationship in (1), or equivalently (2), may be written as

$$\underline{a}^T \underline{\zeta} Y(z) = \underline{b}^T \underline{\zeta} X(z) \tag{8}$$

where

$$\underline{a} = [1 \; a_1 . . . . . . \; a_n]^T$$

$$\underline{b} = [b_0 b_1 \; . . . . . \; b_n]^T$$

$$\underline{\zeta} = [1 \; z^{-1} \; . . . . . \; z^{-n}]^T$$

On the other hand the measurement signals have the following representation. Consider the output measurement signal $y_i(k)$; its transform is

$$\begin{aligned} Y_i(z) &= M_i(z)\, Y(z) \\ &= \prod_{\ell=1}^{i} \frac{1-P_\ell z^{-1}}{1-Q_\ell z^{-1}}\, y(z) \\ &= \frac{1}{D(z)} \prod_{\ell=1}^{i} (1-P_\ell z^{-1}) \prod_{\ell=i+1}^{n} (1-Q_\ell z^{-1}) Y(z) \\ &= \frac{1}{D(z)} \, [c_{oi} \; c_{1i} \; . \; . \; c_{ni}] \, \underline{\zeta} \, Y(z) \end{aligned} \tag{9}$$

where

$$D(z) = \prod_{\ell=1}^{n} (1 - Q_\ell z^{-1}) \tag{10}$$

and the numbers c_{ji} are the same as defined in equation (6). Now the output measurement vectors may be written as

$$\underline{Y}(z) = [Y_0(z) \; . \; . \; . \; Y_n(z)]^T = \frac{1}{D(z)} \, C^T \underline{\zeta} \, Y(z) \tag{11}$$

where the (n+1) x (n+1) matrix $C = [c_{ji}]$ is given by the numbers c_{ij} defined above. In a like manner, we have

$$\underline{X}(z) = [X_0(z) \; . \; . \; . \; X_n(z)]^T = \frac{1}{D(z)} \, C^T \underline{\zeta} \, X(z) \tag{12}$$

We can now state and prove the measurement filter theorem.

Theorem - If the signals y(k) and x(k) satisfy (8) for some parameter vector ($\underline{a}$, $\underline{b}$), then the measurement vectors satisfy the orthogonality condition

$$[\underline{\alpha}^T \; \underline{\beta}^T] \begin{bmatrix} \underline{y}(k) \\ -\underline{x}(k) \end{bmatrix} = 0 \quad \text{for all } k \tag{13}$$

where

$$\underline{\alpha} = C^{-1}\underline{a} \tag{14}$$

$$\underline{\beta} = C^{-1}\underline{b} \tag{15}$$

Proof. The matrix C can be shown to be nonsingular. Therefore we can rewrite

(8) as

$$(C^{-1}\underline{a})^T\ C^T\underline{\xi}\ Y(z) \ = \ (C^{-1}\underline{b})\ C^T\underline{\xi}\ X(z)$$

or

$$\underline{\alpha}^T\ \frac{1}{d(z)}\ C^T\ \underline{\xi}Y(z)\ -\ \underline{\beta}^T\ \frac{1}{d(z)}\ C^T\ \underline{\xi}X(z)\ =\ 0$$

Upon substituting (11) and (12) this equation yields

$$[\underline{\alpha}^T\quad \underline{\beta}^T]\ \begin{bmatrix} \underline{Y}(z) \\ -\underline{X}(z) \end{bmatrix}\ =\ 0 \tag{16}$$

The result sought by the theorem is obtained immediately upon taking the inverse transform.

2.2 APPLICATION EXAMPLES

EXAMPLE 1

As a first example of the identification routine, data from the transfer function

$$H_4(s) = \frac{(10^6)}{s+7(10^6)} + \frac{3(10^7)}{s+7(10^7)} + \frac{(10^6)}{s+(10+j25)10^6} + \frac{(10^6)}{s+(10-j25)10^6}$$

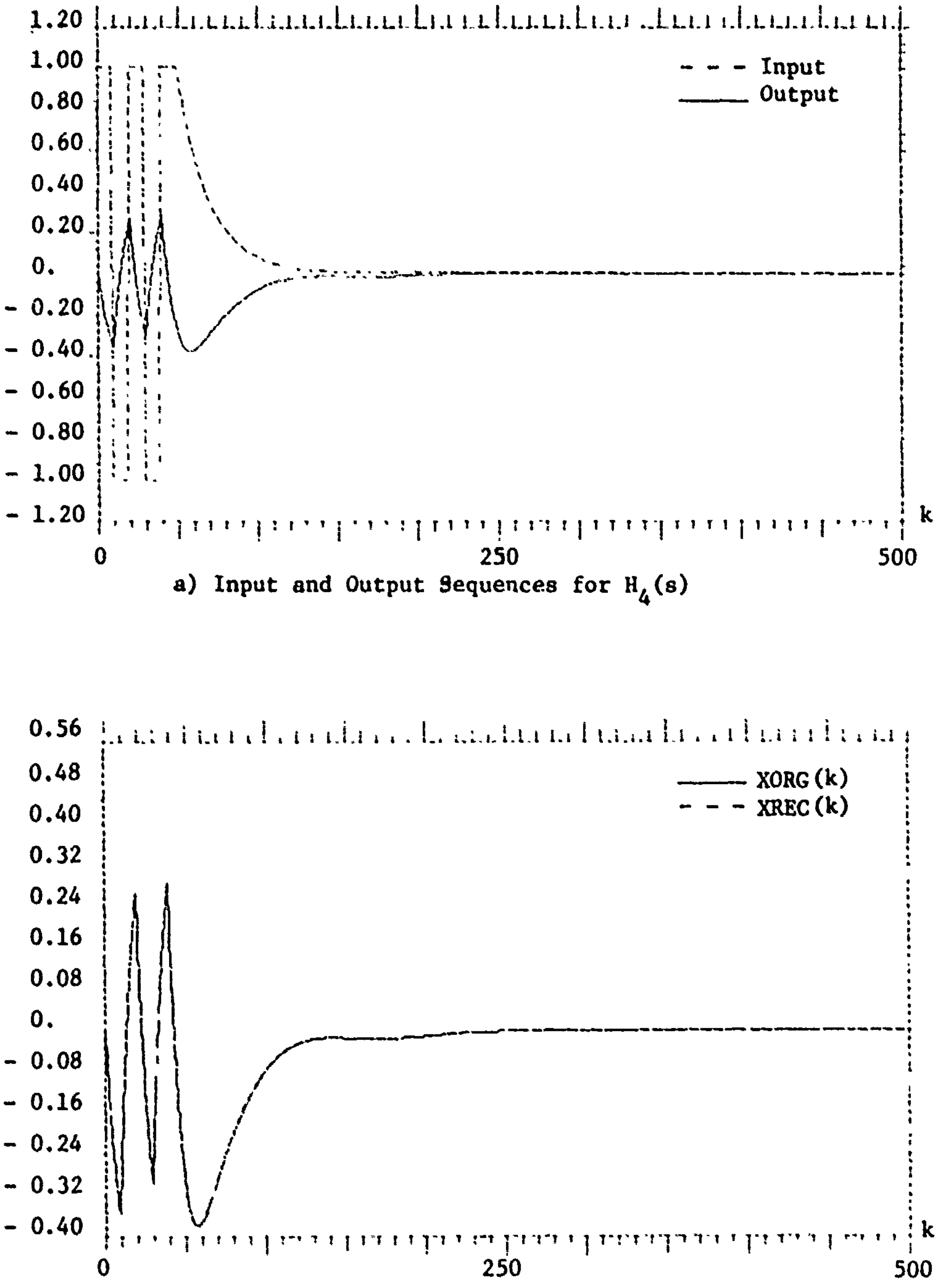
was obtained. The input driving function was a square-wave followed by a decaying exponential. Two cycles of square wave with period 0.05μsec were used, followed by the decaying exponential with time constant approximately equal to 0.05μsec. This input was selected based on apriori knowledge of the network behavior. The spectral content of this input should supply a sufficient amount of energy to the fast mode ($s = -7 \times 10^7$) and to the slow and oscillatory modes for accurate identification.

Five hundred points of data were used (MP1=500) with a sampling interval $\Delta = 0.002\mu\text{sec}$. The option IREM =1 was used in this example because direct transmission could not be assumed. A fourth order model* (N=4) was desired for this network. The identified poles and residues of the model are given below, together with the actual ($H_4(s)$) poles and residues.

$H_4(s)$ Poles	Identified Poles	$H_4(s)$ Residues	Identified Residues
$-7.\text{x}10^6$	$-7.0\text{x}10^6$	$1.0(10^6)$	$1.0(10^6)$
$-7.\text{x}10^7$	$-7.0\text{x}10^7$	$30.0(10^6)$	$30.0(10^6)$
$-(10.+j25)10^6$	$-(10.0+j25.0)10^6$	$1.0(10^6)$	$1.0(10^6)$
$-(10.-j25)10^6$	$-(10.0-j25.0)10^6$	$1.0(10^6)$	$1.0(10^6)$

As seen, the identification of the transfer function was very accurate, and the corresponding mean-square and root-mean square errors for this identification are both 0.0%. Plots of the input-output data and the actual and model responses are shown in Fig. 2.

* A rational transfer function with an nth degree denominator is referred here as being of nth order.



a) Input and Output Sequences for $H_4(s)$

b) Model Response and Actual Response

Fig. 2 . Fourth Order Model Identification of $H_4(s)$

Example 2

We examine the applicability of the identification technique to responses obtained from a wide-band transistor amplifier circuit. The schematic and equivalent model of the circuit are shown in Fig. 3.

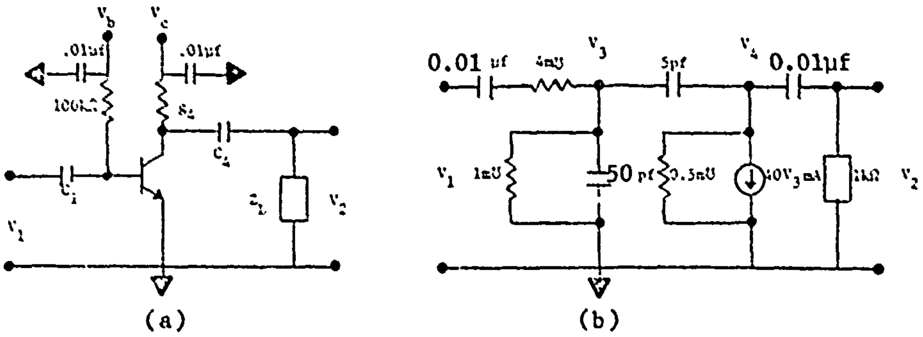


Fig. 3. (a) Schematic of common emitter amplifier circuit
(b) Equivalent circuit model

As shown in Appendix C, the network transfer function is

$$H(s) = \frac{V_2(s)}{V_1(s)} = \frac{8(10^7)\, s^2 (s-8000\ (10^6))}{(s+.033(10^6))(s+.080(10^6))(s+25.2(10^6))(s+1205.1(10^6))} \qquad (17)$$

The network function can be identified successfully only by performing separate tests in three different frequency regions:

(i) Low frequency region (L)
(ii) Mid to High frequency transition (MH)
(iii) High Frequency region (H)

A discussion of these three regions is given in Appendix C. Here we shall focus primarily on the results of identification.

(i) Low frequency region

An adequate low frequency description of eq. (17) is given by *

$$H_L(s) = - \frac{(21.04)\ s^2}{(s+.033(10^6))(s+.080(10^6))} \qquad (18)$$

It is therefore desirable that we seek a second-order (N=2) model of the network given by eq. (17). The input used is a single triangular pulse of duration 125μsec. One thousand points (MP1=1000) of input-output responses,

* In practical applications, such approximations will of course not be available. But the designer, or even the test engineer, frequently has some idea of the critical frequencies of the system.

sampled at $\Delta = 0.25\mu s$ are used for modeling. The option IREM = 0 was used because our low-frequency model will exhibit direct transmission.

The computer program IGRAM yields the following low-frequency model:

$$\hat{H}_L(s) = -\ 20.125\ \frac{(s-0.0015(10^6))(s+0.0012(10^6))}{(s+0.034(10^6))(s+0.075(10^6))} \qquad (19)$$

Comparison of the identified model, eq. (19), with our low frequency approximation (eq. (18)) shows close agreement. The rms error between the measured network time response and the model response is 1.206%. Plots of the input-output signals and of the actual and model responses are given in Fig. 4.

ii) Mid to High frequency transition

As discussed in Appendix C, an adequate mid to high frequency transition description of eq. (17) is given by

$$H_{MH}(s) = -\ \frac{531.1\ (10^6)}{(s\ +\ 25.2(10^6))} \qquad (20)$$

which is a single pole function. Thus, we will attempt to model the circuit with a first-order transfer function (N=1). Since this approximate description (eq. (20)) does not exhibit direct transmission, IREM ≠ 0 should be used in the program. For this identification, IREM = 1 was selected. For reasons mentioned in Appendix C, bias was assumed present (IBIAS = 1) on the data. Five hundred points (MP1 = 500) of input-output signals, sampled at $\Delta = 0.01\mu sec$, were used. The excitation used was a narrow band signal with a center frequency near the critical frequency $(s=-25.2(10^6))$.

The computer program IGRAM yields the following mid to high frequency transition model:

$$\hat{H}_{MH}(s) = \frac{520.0\ (10^6)}{(s+24.92(10^6))} \qquad (21)$$

The rms error between this model and the actual network respon (obtained from the original 4th order transfer function) is 1.992%. A graphical comparison is given in Fig. 5.

iii) High frequency region

An approximate high frequency description of eq. (17) is shown (Appendix C) to be

$$H_H(s) = \frac{8(10^7)\ (s-8000(10^6))}{(s+25.2(10^6))(s+1205.1(10^6))} \qquad (22)$$

which is a two-pole function. We will therefore attempt to model the circuit with a second-order transfer function (N=2). Direct transmission cannot be assumed, so that IREM = 1 was used for the model. Five hundred points (MP1 = 500) of input-output signals, sampled at Δ = 0.00025μsec were used. A narrow-band signal with a center frequency of 6300 Mrad/sec was used for network excitation.

The computer program yields the following high frequency model:

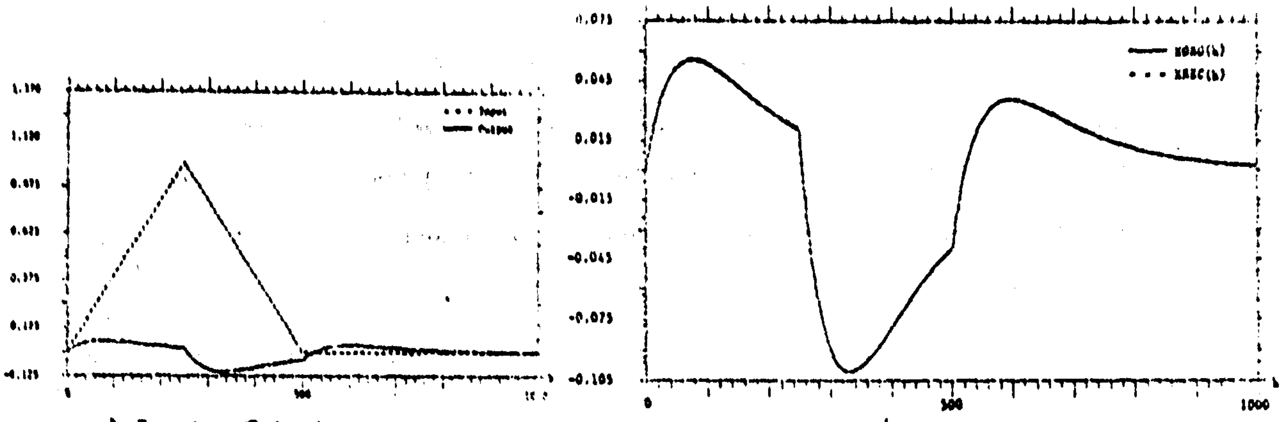
$$\hat{H}_H(s) = \frac{2.79(10^6)\ ((s-19060(10^6))}{(s+25.7(10^6))(s+1139.5(10^6))} \qquad (23)$$

The rms error between this model and the actual network response is 0.919%. A graphical comparison is given in Fig. 6.

We have now obtained a description of the networ' behavior in the three frequency regions of interest. The models obtained may be pieced together in such a way as to synthesize the overall behavior of the system. Omitting the details involved, the following model may be obtained from eq's. (19), (21) and (23):

$$\hat{H}(s) = \frac{62.2(10^6)\ (s-.0015(10^6))(s+.0012(10^6))(s-199060(10^6))}{(s+.034(10^6))(s+.075(10^6))(s+24.9\ (10^6))(s+1139.5(10^6))}$$

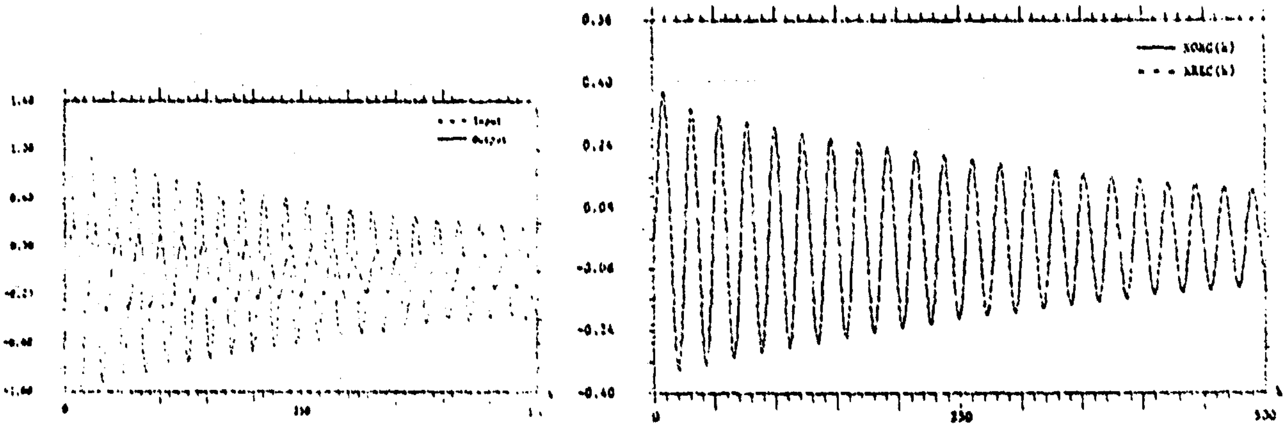
Comparison of this model with eq. (17) shows favorable agreement.



a) Input - Output signals for $H_L(s)$

b) Comparison of model and network responses

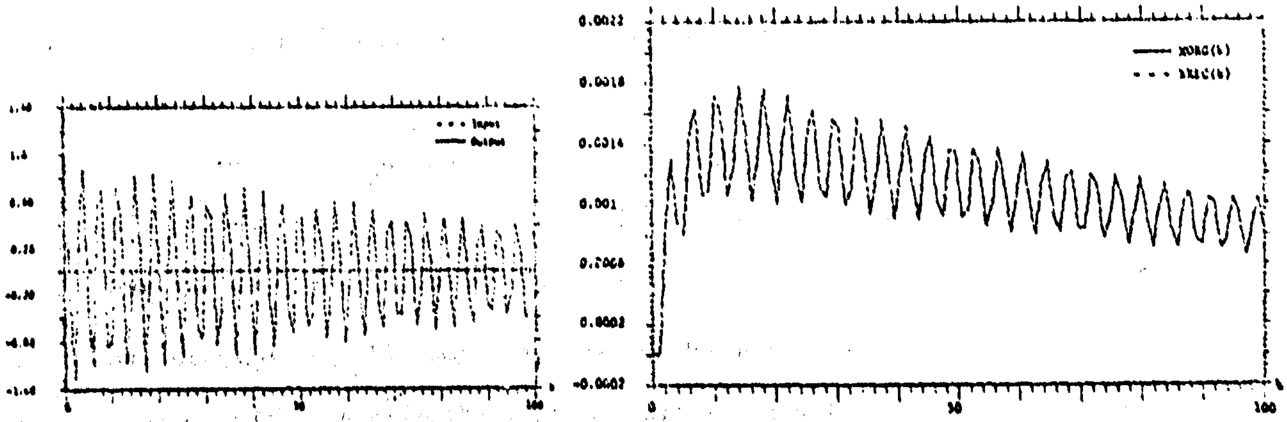
Fig. 4. Second Order Model Identification of $H_L(s)$



a) Input - Output signals for $H_{MH}(s)$

b) Comparison of model and network responses

Fig. 5. First Order Model Identification of $H_{MH}(s)$



a) Partial input - output signal for $H_H(s)$

b) Comparison of model and network responses

Fig. 6. Second Order Model Identification of $H_H(s)$

2.3 Program Description

This FORTRAN IV program determines a linear model (transfer function) of a network from recorded laboratory responses. The linear model is obtained via the pencil-of-functions method discussed in Section 2.1. The program has certain features which are discussed below.

Network modeling involves the determination of the coefficients α_i and β_i of a rational transfer function of the form

$$H(s) = \frac{\beta_o + \beta_1 s^1 + \ldots + \beta_n s^n}{\alpha_o + \alpha_1 s^1 + \ldots + \alpha_n s^n} \tag{17}$$

such that the output of this model to a given input will approximate the actual network output to the same input. Equivalently* in discrete time [17] we wish to determine the coefficients a_i and b_i of a function of the form

$$H(z) = \frac{b_o + b_1 z^{-1} + \ldots + b_n z^{-n}}{a_o + a_1 z^{-1} + \ldots + a_n z^{-n}} \tag{18}$$

If the network under study is assumed to have direct transmission, the numerator coefficient b_o is nonzero. This choice of model structure is implemented by setting IREM = 0. When direct transmission cannot be assumed (i.e., it is known on physical grounds that the impulse response of the network will not contain an impulse), then b_o should be set to zero. This is accomplished with IREM≠0. For example, if IREM=1, the coefficient b_0 in equation (18) is set to zero; for IREM=2 the coefficients b_0 and b_1 are set to zero. It is recommended to use IREM=1 whenever direct transmission cannot be assumed.

All calculations are performed in discrete time; finally H(z) is transformed by means of a pulse invariant transformation*(IZTS=2) to the corresponding continuous time model H(s). After modeling has been accomplished, the

* See Appendix D

Normalized mean-square error (and its square root) comparing the model and actual network responses are calculated (subroutine ERROR). These errors are calculated as shown below.

$$\text{N.M.S.E.} = \frac{\sum\limits_{k} [x(k) - x_{\text{model}}(k)]^2}{\sum\limits_{k} x^2(k)}$$

$$\text{R.N.M.S.E.} = \sqrt{\text{N.M.S.E.}}$$

Another feature of the program is the capability for bias-removal from the recorded laboratory responses (IBIAS=1). This feature allows consideration for bias that may have been introduced through the laboratory measurement system to the recorded output-input data.

Finally, a plot option (IPLT) is available. When IPLT=1, two sets of plots are given. The first shows the original output-input data measured from the network. The second plot contains the original network response and the identified linear model response. This plot allows visual inspection of the closeness of the model fit to the actual (desired) response.

To enable the test engineer to effectively use the program, a description of the input data cards is given below.

INPUT DESCRIPTION

CARD #1 The first card is a title card. Columns 1 through 80 are available for an alpha-numeric title.

CARD #2 Option card containing three variables

Variable Name (Format)	Description	Columns
N(I5)	Order of the system	1-5
MP1(I5)	Number of data points (output-input data)	6-10

IPLT(I5)	Plotter option;
	IPLT = 0 no plots
	IPLT = 1 plots on line printer
		11-15

CARD #3 through CARD [2+NOUT]

NOUT = [(MP1+7)/8], where [X] is the truncated value of X.

The output data is entered on these cards in 8F10.0 fields.

CARD #[3+NOUT] through CARD [2+NOUT+NIN]

NIN = [(MP1+7)/8], where [X] is the truncated value of X.

The input data is entered on these cards in 8F10.0 fields.

*CARD #[3+NOUT+NIN] Second option card containing six variables.

Variable Name (Format)	Description	Columns
N(I5)	Order of the system	1-5
MP1(I5)	Number of data points (output-input data)	6-10
ISKIP(I5)	This variable determines the sequence of points plotted on the printer. If ISKIP = 1 every data point is plotted, and if ISKIP = 5 every fifth point is plotted, etc.	11-15
IREM(I5)	Variable used to specify model structure for the identified system. If direct transmission is assumed, IREM=0. For IREM=m, the first m terms (in ascending order) of the model numerator polynomial are set to zero. It is recommended IREM=1 when direct transmission cannot be assumed.	16-20

*At first glance, this card may seem partially repetitious with CARD #2. However, when multiple identification runs on the same output-input data are desired, then more than one such option card may be placed here, with the option variables changed as desired (for instance, a run on only part of the output-input sequence may at times be needed).

IBIAS(I5)	Bias-removal option IBIAS = 0 no bias is assumed present on the output-input data. IBIAS = 1 bias, assumed to have been introduced by the measurement system, is removed before identification is performed.	21-25
DELTA(F5.0)	Sampling interval	26-30

<u>END OF FILE CARD</u>

A listing of the FORTRAN programs used is given in Appendix A .

III. COMPUTER ROUTINE FOR HIGH PRECISION INVERSION OF SECOND ORDER VOLTERRA RESIDUE MATRICES

The determination of the residues of the quadratic TF, $H_2(s_1,s_2)$, involves the solution of a set of linear equations. Unfortunately, the number of equations involved are large, for example 12 [2] even for a modest single pole-pair situation (i.e., where the linear TF has two distinct poles). Solution of these equations can lead to computational errors unless extreme caution is exercised in the inversion of the associated matrix. In fact the problem is further aggrevated in cases where the system is wide band, i.e., when the poles of $H_1(s)$ are spaced several decades apart. In such situations, the poles of $H_2(s_1,s_2)$ involve sums and differences of the linear TF poles which can result in precariously close values. For example, if

$$\lambda_1 = 50 \text{ radians/s}$$

$$\lambda_2 = 50 \text{ M radians/s}$$

then

$$\lambda_1 + \lambda_2 = 50.00005 \text{ Mradians/s}$$

$$\lambda_1 - \lambda_2 = 49.99995 \text{ Mradians/s}$$

This in turn causes the associated columns of the coefficient matrix corresponding to these poles to be almost scalar multiples of each other. The matrix thus becomes nearly singular, or highly ill-conditioned to invert.

The program presented in this section is designed to deal with such wideband cases, and more generally, to invert ill-conditioned matrices where ever they may arise. It is hoped that by mastering the various capabilities of this routine the analyst can cope with almost all situations of practical interest.

The program possesses the following features which enable high-precision inversion:

Adaptive Scaling
Application of Perturbation Theory to Ill-Conditioned Matrices
Iterative Correction

Before discussing each of these in detail, it is useful to define the term "ill-conditioned" matrix. We will call a matrix ill-conditioned if (a) the rows (or columns) of the matrix are nearly dependent, (b) "small" changes in one or more entries of the matrix result in large changes in its inverse, or (c) the nonzero entries of the matrix differ widely by several orders of magnitude (and remain so even after appropriate scaling has been performed) [18], [19], [20]. Note, the above conditions are not mutually exclusive.

3.1 ADAPTIVE SCALING

In many applications the entries of a matrix differ widely in their respective sizes. For a linear system of equations this situation arises when the values of the (unknown) variables are orders of magnitude different and/or the various equations have right-hand-sides which are orders of magnitude different.

This situation can be remedied in many cases as follows. Denote the matrix of interest as A_o. Then it is possible to factorize A_o as

$$A_o = PA_1 = PAQ \tag{1}$$

where P and Q are suitable diagonal scaling matrices [19]. The following method was developed to obtain the diagonal matrices P and Q, and hence the new matrix, A.

The diagonal entries of matrix P are successively computed from the product of all "significant" terms in the successive rows. The term "significant" can be specified by the user (in the examples presented here any entry greater than 15 orders of magnitude below the largest entry in the row of interest was considered significant); a default value of 15 orders of magnitude is assumed. Then, the P_{ii}th entry of the diagonal matrix P is computed as the (n_i)th root of the magnitude of the aforementioned product, where n_i is the number of terms in the product.[1]

[1] The scaling of the ith row may be stated mathematically as follows; let

$$\alpha_i = \underset{j}{\mathrm{Max}}\,\mathrm{ABS}(A_o)_{ij} \quad \text{[largest entry of ith row]}$$

$$\beta_i = \alpha_i * 10^{-m} \quad \text{[threshold for ith row (m choosen by the user)]}$$

$$(A_o)_{ij} : \mathrm{ABS}(A_o)_{ij} > \beta_i \quad \text{[qualifying entries of ith row]}$$

$$n_i = \text{number of qualifying entries in ith row}$$

Then

$$P_{ii} = \left\{ \prod_{\substack{\text{qualifying} \\ \text{entries}}} (A_o)_{ij} \right\}^{\frac{1}{n_i}}$$

At this point, we have factorized matrix A_o into two matrices,i.e., $A_o = P * A_1$, where the entries of A_1 are specified as follows:

$$(A_1)_{ij} = (A_o)_{ij}/P_{ii}, \qquad j = 1, \ldots, n \qquad (2)$$

It should be noted that in certain cases, the above row scaling will suffice, and further scaling may not be necessary. However, in general, the above process may be repeated, this time utilizing column scaling. Specifically, the column scaling involves factorizing A_1 such that $A_1 = A * Q$, where Q is diagonal. The entries of A are obtained as

$$(A)_{ij} = (A_1)_{ij}/Q_{jj}, \qquad i = 1, \ldots, n \qquad (3)$$

The entries of the scaling matrix Q are chosen in the same manner as those of P, except that columns rather than rows (of A_1) are examined.

Utilizing this technique, the desired inverse is seen to be

$$A_o^{-1} = Q^{-1} A^{-1} P^{-1}. \qquad (4)$$

Example 1

$$A_o = \begin{bmatrix} 0.10000000\text{ E+03} & 0.20000000\text{ E-04} & 0.29999999\text{ E-01} \\ 0.19999989\text{ E+06} & 0.40000000\text{ E-01} & 0.60000000\text{ E+02} \\ -0.10000000\text{ E+10} & 0.10000000\text{ E+03} & 0.0 \end{bmatrix}$$

Inspection of matrix A_o shows that its entries differ widely in relative magnitudes; in fact there is a difference of fourteen orders of magnitude. Therefore, scaling can be used, yielding the following: $A_o = PA_1 = PAQ$, where

$$P = \begin{bmatrix} 0.391486768\text{E-01} & 0.0 & 0.0 \\ 0.0 & 0.78297339\text{E+02} & 0.0 \\ 0.0 & 0.0 & 0.31622776\text{E+06} \end{bmatrix}$$

Row Scaled Matrix, $$A_1 = \begin{bmatrix} 0.25543647\text{E+04} & 0.51087295\text{E-03} & 0.76630943\text{E+00} \\ 0.25543639\text{E+04} & 0.51087304\text{E-03} & 0.76630956\text{E+00} \\ -0.31622776\text{E+04} & 0.31622776\text{E-03} & 0.0 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.27427647\text{E+04} & 0.0 & 0.0 \\ 0.0 & 0.43538684\text{E-03} & 0.0 \\ 0.0 & 0.0 & 0.76630949\text{E+00} \end{bmatrix}$$

Row and column scaled matrix,

$$A = \begin{bmatrix} 0.93131018E+00 & 0.11733771E+01 & 0.999999910E+00 \\ 0.93130987E+00 & 0.11733773E+01 & 0.10000001E+01 \\ -0.11529525E+01 & 0.726314477E+00 & 0.0 \end{bmatrix}$$

Now, $A_o^{-1} = Q^{-1} A^{-1} P^{-1}$, and the inversions yield:

$$A_o^{-1} = \begin{bmatrix} 0.20000000E+05 & -0.10000000E+02 & -0.54977761E-18 \\ 0.20000000E+12 & -0.10000000E+09 & 0.99999999E-02 \\ -0.19999996E+09 & 0.10000000E+06 & -0.66666666E-05 \end{bmatrix}$$

The product $A_o(A_o)^{-1}$ is given by

$$A_o(A_o)^{-1} = \begin{bmatrix} 0.10000000E+01 & -0.17053025E-12 & 0.26469779E-22 \\ -0.95367431E-06 & 0.10000000E+01 & 0.54210108E-19 \\ 0.0 & -0.95367431E-06 & 0.10000000E+01 \end{bmatrix}$$

Checking Product of A_{orig} and Its Computed Inverse

After having computed the inverse, it is desirable to check the accuracy of inversion. The obvious way to do so is to obtain the product of A_o and A_o^{-1} and to examine how close it is to the unit matrix. However, in view of the finite word-length of the computer, this product must be computed carefully. Whenever scaling methods are employed, the product of A_o and its inverse may be determined in a number of ways. This will be discussed for three different cases: (i) row scaling, (ii) column scaling, (iii) both row and column scaling.

(i) Row Scaling: $A_o = PA$

In this case, $A_o^{-1} = A^{-1} P^{-1}$. Therefore, as a check on the dependability of A_o^{-1}, the product $A_o A_o^{-1}$ would probably be examined as follows:

$$A_o A_o^{-1} = P(A\, A^{-1})\, P^{-1} \qquad (5a)$$

On the other hand,

$$A_o^{-1} A_o = A^{-1}(P^{-1}P)A = A^{-1}A \qquad (5b)$$

Equations (5a) and (5b), while representing the same quantity ($A_o A_o^{-1} = A_o^{-1} A_o$), may not be found equal due to the available comouter accuracy (finite word-length). Example 2 illustrates this point with an extreme case.

EXAMPLE 2:

$$A_o = \begin{bmatrix} 0.10000000\text{E+}03 & 0.20000000\text{E-}04 & 0.29999999\text{E-}01 \\ 0.19999990\text{E+}19 & 0.40000000\text{E+}12 & 0.60000000\text{E+}15 \\ -0.50000000\text{E-}08 & 0.49999999\text{E-}11 & 0.0 \end{bmatrix}$$

Factorizing A_o, we obtain $A_o = PA$, where

$$P = \begin{bmatrix} 0.39148676\text{E-}01 & 0.0 & 0.0 \\ 0.0 & 0.78297339\text{E+}15 & 0.0 \\ 0.0 & 0.0 & 0.15811383\text{E-}09 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.25543647\text{E+}04 & 0.51087295\text{E-}03 & 0.75630943\text{E+}00 \\ 0.25543639\text{E+}04 & 0.51087304\text{E-}03 & 0.76630956\text{E+}00 \\ -0.31622776\text{E+}02 & 0.31622776\text{E-}01 & 0.0 \end{bmatrix}$$

Now, matrix A is inverted, yielding

$$A^{-1} = \begin{bmatrix} 0.78297352\text{E+}03 & -0.78297339\text{E+}03 & -0.22097009\text{E-}14 \\ 0.78297352\text{E+}06 & -0.78297339\text{E+}06 & 0.316227766\text{E+}02 \\ -0.26104324\text{E+}07 & 0.26104333\text{E+}07 & -0.210818510\text{E-}01 \end{bmatrix}$$

The product AA^{-1} is computed as

$$\begin{bmatrix} 0.99999999\text{E+}00 & 0.23283064\text{E-}09 & -0.43368086\text{E-}17 \\ -0.23283064\text{E-}09 & 0.10000000\text{E+}01 & -0.43368086\text{E-}17 \\ -0.45475735\text{E-}11 & 0.272848411\text{E-}11 & 0.99999999\text{E+}00 \end{bmatrix}$$

The required inverse, A_o^{-1}, is computed as $A_o^{-1} = A^{-1}P^{-1}$. Once A_o^{-1} has been computed, we can obtain the product of A_o and A_o^{-1} by use of equation (5a) or (5b). Utilizing Eq. (5a), yields

$$A_o A_o^{-1} = P(AA^{-1})P^{-1} \approx \begin{bmatrix} 0.99999999E+00 & 0.0 & -0.21475700E-09 \\ 0.0 & 0.99999999E+00 & -0.42951393E+07 \\ 0.0 & 0.0 & 0.99999999E+00 \end{bmatrix}$$

Comparison of this matrix with the unit matrix (the (2,3) entry in particular) would yield the fauity conclusion that an accurate inverse has not been obtained. If, on the other hand, Eq (5b) is used as a check, we find that

$$A_o^{-1} A_o = A^{-1}(P^{-1}P)A = A^{-1}A = \begin{bmatrix} 0.99999999E+00 & -0.69876856E-16 & -0.17053025E-12 \\ -0.29795324E-06 & 0.99999999E+00 & -0.10186340E-09 \\ 0.25428985E-05 & 0.36878730E-12 & 0.10000000E+01 \end{bmatrix}$$

Comparison of this matrix with the unit matrix would be favorable. Thus, in the case of row scaling, equation (5b) must be used as a check on the product A_o and A_o^{-1} due to the finite computer word-length . Use of this equation avoids the possible problem (present in Eq. 5a) that the matrices P, (AA^{-1}) and P^{-1} may not be compatible for multiplication, as exemplified above. To reiterate, in the case of row-scaling the product of A_o and A_o^{-1} must be computed as $A_o^{-1}A_o = A^{-1}A$, where A is the scaled matrix.

ii) Column Scaling: In a similar manner we will show that the product AA^{-1} must be used as a check on the goodness of A_o^{-1} in the case of column scaling. This can be seen from a comparison of the following equations, where $A_o = AQ$ and $A_o^{-1} = Q^{-1}A$.

$$A_o^{-1}A_o = Q^{-1}(A^{-1}A)Q \tag{6a}$$

$$A_oA_o^{-1} = A(QQ^{-1})A^{-1} = AA^{-1} \tag{6b}$$

Again due to finite word length in the computer, equation (6b) should be used to verify the goodness of the inverse when column-scal.ng is performed.

iii) Row and Column Scaling: In this case, matrix A_o is factorized as $A_o = PAQ$, with the required inverse, $A_o^{-1} = Q^{-1}A^{-1}P^{-1}$. The decision to use $A_oA_o^{-1}$ or $A_o^{-1}A_o$ as a check towaru the accuracy of the inverse obtained may be arrived at as follows. The equations representing $A_oA_o^{-1}$ and $A_o^{-1}A_o$ are

$$A_oA_o^{-1} = PA(QQ^{-1})A^{-1}P^{-1} = P(AA^{-1})P^{-1} \qquad (7a)$$

and

$$A_o^{-1}A_o = Q^{-1}A^{-1}(P^{-1}P)AQ = Q^{-1}(A^{-1}A)Q \qquad (7b)$$

The difference in magnitude between the largest and smallest entries in the diagonal matrices P and Q is calculated. These two differences are compared, and the matrix with the largest difference is attempted to be eliminated (either P or Q). Thus, if the entries of P are more incompatible for multiplication than those of Q, equation (7b) would be used as a check rather than equation (7a). An example will help to clarify this procedure.

EXAMPLE 3:

$$A_o = \begin{bmatrix} 0.10000000E+03 & 0.20000000E-04 & 0.29999999E-01 \\ 0.19999990E+19 & 0.40000000E+12 & 0.60000000E+15 \\ -0.50000000E-08 & 0.49999999E-11 & 0.0 \end{bmatrix}$$

The matrix is now expressed as $A_o = PAQ$, where

$$P = \begin{bmatrix} 0.39148676E-01 & 0 & 0.0 \\ 0.0 & 0.78297339E+15 & 0.0 \\ 0.0 & 0.0 & 0.15811388E-09 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.59091075E+03 & 0.0 & 0.0 \\ 0.0 & 0.20208866E-02 & 0.0 \\ 0.0 & 0.0 & 0.76630949E+00 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.43227589E+01 & 0.25279643E+00 & 0.99999991E+00 \\ 0.43227575E+01 & 0.25279647E+00 & 0.10000000E+01 \\ -0.535153177E-01 & 0.15647971E+02 & 0.0 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 0.46266748E+06 & -0.46266740E+06 & -0.13086035E-11 \\ 0.15823007E+04 & -0.15823005E+04 & 0.63906048E-01 \\ -0.20003991E+07 & 0.20003998E+07 & -0.16155222E-01 \end{bmatrix}$$

Comparing the magnitude difference between the largest and smallest entries of P and Q, it is found that in P, the difference is approximately 10^{24}, while in Q the difference is approximately 10^{5}. Therefore, Eq. (7b) should be used, in order that multiplication involving P and P^{-1} is avoided. Performing the required multiplication indicated in Eq. (7b), we obtain

$$A_o^{-1}A_o = Q^{-1}(A^{-1}A)Q = \begin{bmatrix} 0.99999999E+00 & -0.70030334E-16 & -0.56613984E-13 \\ -0.23651533E-06 & 0.99999999E+00 & -0.43109444E-10 \\ 0.89538983E-06 & 0.15661482E-12 & 0.10000000E+01 \end{bmatrix}$$

3.2 PERTURBATION

In some applications, the given matrix may be ill-conditioned to the point that the scaling method described above will not allow inversion of the matrix to the desired precision. In this case, application of perturbation theory to the scaled matrix, A, may be helpful. Several methods will be described.

A. Diagonal Perturbation

The first method consists of forming a new matrix [18]

$$C = A + \varepsilon D \qquad (8)$$

where A is the scaled version of the original matrix and D is a diagonal matrix whose entries can be taken as those of the diagonal of A. The multiplier, ε, is chosen to be suitably small (more will be said about this choice later) such that C is invertible (allowing for the available computer accuracy). We can now write $A = C - \varepsilon D$ so that we have parameterized A in terms of the small parameter, ε. Since $A = A(\varepsilon)$ is an analytic function of ε, its inverse is also an analytic function of ε; therefore, a power-series may be written. Indeed,

$$\begin{aligned} A^{-1} &= (C - \varepsilon D)^{-1} \\ &= (1 - \varepsilon C^{-1}D)^{-1}C^{-1} \\ &= [C^{-1} + \varepsilon C^{-1}DC^{-1} + \varepsilon^2(C^{-1}D)^2C^{-1} + \ldots] \end{aligned} \tag{9}$$

Thus, using C^{-1}, D, and ε, the inverse of A may be computed. An advantage of this method is that it can be performed without any visual inspection of the given matrix. This is so because the diagonal matrix D is specified automatically. A disadvantage worth noting is that the required inverse, A^{-1}, is represented as an infinite series of matrices and cannot be expressed in closed form. Thus, an exact representation of A^{-1} cannot be obtained, although it can be approximated to the required accuracy by computing and summing enough terms of the series.

We now return to the matter of choosing a suitable value of ε. Clearly, the smaller the values of ε, the faster will be the convergence of the series (9). On the other hand, ε must be large enough so that it results in adequate perturbation, i.e., such that C is invertable with the available computer accuracy. It is useful to note that a theoretical upper bound on ε can be obtained; indeed for the series (9) to converge it is necessary that $|\varepsilon|$ be less than 1.0/|largest eigenvalue of $C^{-1}D$|. [19], [20].

B. Single-entry Perturbation

An alternate perturbation method consists again of forming a new matrix

$$C = A + \varepsilon D$$

where we now restrict the perturbation matrix D to have only one non-zero entry, again on the diagonal. Without loss of generality, this non-zero entry can be taken to be unity. Then, the matrix D has rank-one, trace-one and hence can be written as

$$D = XY^T \tag{10}$$

Thus

$$\begin{aligned} A^{-1} &= (C - \varepsilon XY^T)^{-1} \\ &= [I - \varepsilon(C^{-1}X)Y^T]^{-1}C^{-1} \\ &= I + \varepsilon C^{-1}X[1 + \varepsilon Y^TC^{-1}X + (\varepsilon Y^TC^{-1}X)^2 + \ldots](Y^TC^{-1}) \end{aligned} \tag{11}$$

Now, the quantity in brackets in equation (11) is a power-series involving scalar terms. Therefore, equation (11) can be expressed as

$$A^{-1} = I + \varepsilon C^{-1}X\left[\frac{1}{1 - \varepsilon Y^Tc^{-1}X}\right](Y^TC^{-1}) \tag{12}$$

It can be shown that the series converges for all ε. It is, however, best to choose $\varepsilon < \frac{1}{Y^TC^{-1}X}$. Thus, equation (12) represents an exact solution for A^{-1}. A disadvantage of this method is that visual inspection of the matrix, A, is necessary to determine the row(s) and/or column(s) causing difficulty in the inversion process, and thus a suitable entry to perturb. An illustrative example of this method is given below.

EXAMPLE 4:

$$A_o = \begin{bmatrix} 0.10000000E+03 & 0.20000000E-04 & 0.29999999E-01 \\ 0.19999989E+06 & 0.40000000E-01 & 0.60000000E+02 \\ -0.10000000E+10 & 0.10000000E+03 & 0.0 \end{bmatrix}$$

A visual inspection of this matrix shows that the first and second rows are nearly identical to within a multiplicative factor of 2.0 x 10^3. Therefore, suitable diagonal entries for perturbation would be either the (1,1) or (2,2) entries. The (2,2) entry was selected for perturbation, and through iteration, a suitable value of ε was found to be ε = 1. x 10^{-10}. Scaling was first performed, and then the perturbation yielding:

$$A=C-\varepsilon D=\begin{bmatrix} 0.9313I018E+00 & 0.11733771E+01 & 0.99999991E+00 \\ 0.93130987E+00 & 0.11733773E+01 & 0.10000000E+01 \\ -0.11529525E+01 & 0.72631447E+00 & 0.0 \end{bmatrix} -10^{-10}\begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.173377 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Note that D can be expressed as $D = XY^T$ so that

$$D = \begin{bmatrix} 0.0 \\ 1.173377 \\ 0.0 \end{bmatrix} \begin{bmatrix} 0.0 & 1.0 & 0.0 \end{bmatrix}$$

The inverse of the original matrix can now be computed as discussed:

$$A_o^{-1} = \begin{bmatrix} 0.20000000E+05 & -0.10000000E+02 & 0.52504835E-18 \\ 0.20000000E+12 & -0.10000000E+09 & 0.10000000E-01 \\ -0.19999996E+09 & 0.10000000E+06 & -0.66666666E-05 \end{bmatrix}$$

This yields the product

$$A_oA_o^{-1} = \begin{bmatrix} 0.10000000E+01 & -0.51159076E-12 & 0.66174449E-22 \\ -0.95367431E-06 & 0.10000000E+01 & 0.0 \\ 0.0 & -0.95367431E-06 & 0.99999999E+00 \end{bmatrix}$$

C. PERTURBATION: THE LIMITING CASE: In the previous discussions on perturbation, it was shown that matrix A is a function of matrix C and the scalar quantity, ε. That is $A = (C(\varepsilon),\varepsilon)$. Recall also that whereas A was ill-conditioned (for inversion), there were certain values of ε for which the newly formed matrix C could be made well-behaved. Inspection of equation (8) shows that

$$A^{-1} = \lim_{\varepsilon \to o} C^{-1} \tag{13}$$

This observation can be exploited in the following way. Successively small values of ε, say ε_i, are used to form a family of $C = C(\varepsilon_i)$ matrices. The inverse of $C(\varepsilon_i)$ is computed for each value of ε_i used, and the successive inverses are examined. There will exist a region wherein reducing the value of ε from ε_i to ε_{i+1} will have little effect on the entries of C^{-1}. This is shown graphically in Figure 1.

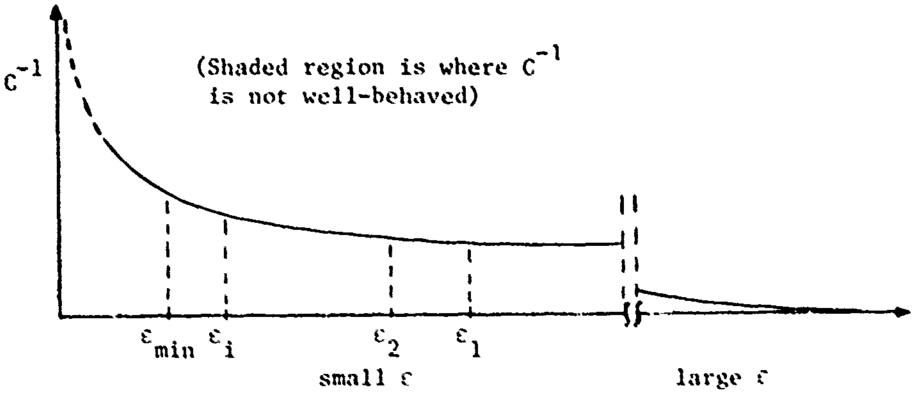


Fig. 1. Effect of small changes in ε on C^{-1}.

In this region, C^{-1} can be taken as an approximation to A^{-1}. As seen in Figure 1, there will exist some ε_{min} for which the inverse of C is well-behaved. The closer the selected value of ε is to this value of ε_{min}, the better will become the approximation $A^{-1} \doteq C^{-1}$. Although this method may only yield an approximation to the actual required inverse, A^{-1}, it may be further refined by use of the method described in the next section. In fact, the iterative correction method (of the next section) may be used in conjuction with all of the methods discussed earlier.

3.3 ITERATIVE CORRECTION

Consider a matrix, X, and assume that its inverse has been computed as $Y \approx X^{-1}$. The iterative correction method (see Fig. 2) consists of forming the product, XY, comparing it with the identity matrix, and improving the computed inverse, Y, by an amount proportional to the error between XY and the unit matrix. To examine the effect of this operation, let

$$Y = X^{-1}(1 + E) \qquad (14)$$

where 1 = identity matrix and E is equal to the difference matrix between XY and I.

$$XY = (I + E) \tag{15}$$

$$E = (XY - I) \tag{16}$$

Now consider the iteration $Y_{\text{improved}} = Y - YE$ [12]. Clearly (17a)

$$\begin{aligned} Y_{\text{improved}} &= Y - YE \\ &= X^{-1}(I + E)(I - E) \\ &= X^{-1}(I - E^2) \end{aligned} \tag{17b}$$

Upon the second iteration,

$$Y_{\text{improved}} = X^{-1}(I + E^4)$$

and so on. This prodecure can be depicted in block-diagram form as in Figure 2.

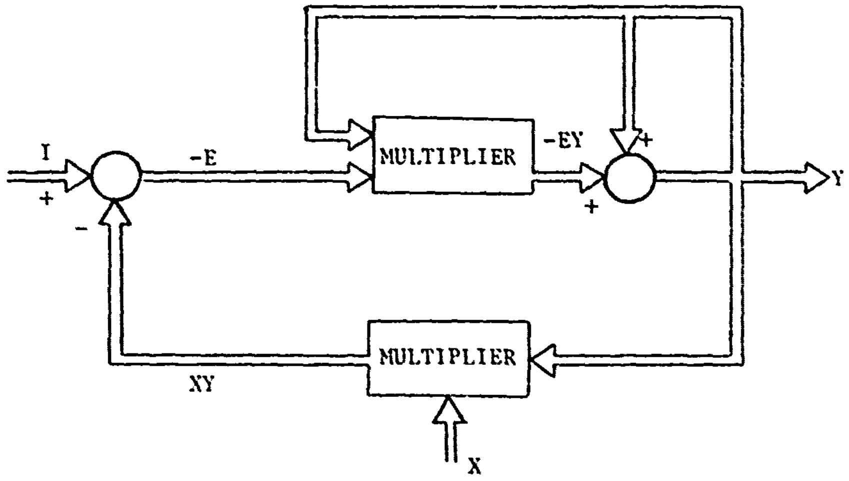


Fig. 2: Block diagram representation of iterative correction method.

The number of iterations to be used may be specified by the user. For this work, n iterations have usually been used, where n denotes the dimension of the matrix in question. Note that a more general version of (17a) is $Y_{\text{improved}} = Y - \beta * YE$ where β is a suitable positive fraction.

<u>EXAMPLE 5</u>: (Effect of iterative correction)

$$A_o = \begin{bmatrix} 0.10000000E{+}03 & 0.20000000E{-}04 & 0.29999999E{-}01 \\ 0.19999999E{+}06 & 0.40000000E{-}01 & 0.60000000E{+}02 \\ -0.10000000E{+}10 & 0.10000000E{+}03 & 0.0 \end{bmatrix}$$

Assume that the inverse matrix has been computed as

$$\tilde{A}_o^{-1} = \begin{bmatrix} 0.19999999E+06 & -0.99999998E+02 & -0.10058593E-16 \\ 0.19999999E+13 & -0.99999998E+09 & 0.99999998E-02 \\ -0.19999999E+10 & 0.99999998E+06 & -0.66666665E-05 \end{bmatrix}$$

so that

$$A_o\tilde{A}_o^{-1} = \begin{bmatrix} 0.99999999E+00 & 0.36379788E-11 & 0.52939549E-22 \\ -0.15258789E-04 & 0.99999999E+00 & 0.0 \\ -0.11718750E-01 & 0.15258789E-04 & 0.99999999E+00 \end{bmatrix}$$

Now, if 1 iteration of the correction method is performed,

$$A_o\tilde{A}_o^{-1} = \begin{bmatrix} 0.10000000E+01 & 0.0 & 0.0 \\ 0.0 & 0.10000000E+01 & -0.54210108E-19 \\ -0.15625000E-01 & 0.0 & 0.99999999E+00 \end{bmatrix}$$

with N(=3) iterations performed,

$$A_o\tilde{A}_o^{-1} = \begin{bmatrix} 0.99999999E+00 & 0.0 & -0.13234889E-22 \\ 0.0 & 0.10000000E+01 & 0.54210108E-19 \\ -0.78125000E-02 & 0.0 & 0.99999999E+00 \end{bmatrix}$$

It can be seen that the product $A_oA_o^{-1}$ is approaching the unit matrix. The worst entry, (3,1), in the product has been reduced to about 60% of its original value in the 3 iterations.

The improved inverse has been computed as

$$\tilde{A}_o^{-1} = \begin{bmatrix} 0.19999999E+06 & -0.99999999E+02 & -0.76657257E-17 \\ 0.19999999E+13 & -0.99999999E+09 & 0.99999999E-02 \\ -0.19999999E+10 & 0.99999999E+06 & -0.66666665E-05 \end{bmatrix}$$

This example has demonstrated the usefulness of the correction method in improving the calculated inverse.

3.4 APPLICATION

Determination of the kernels of the volterra model can be accomplished by identifying the various link transfer functions $H_1(s_1)$, $H_2(s_1,s_2)$, etc. The first step, of course, is to determine $H_1(s_1)$ for the small signal linear case. Next, identification of $H_2(s_1,s_2)$ is attempted. However, it is shown in [1] that the poles of $H_2(s_1,s_2)$ are given as $\gamma_{ik} = \lambda_i + \lambda_k$ where λ_i, λ_k are the poles of the linear transfer function $H_1(s_1)$. The residues at these poles can then be determined by solving a set of linear equations from data obtained for larger amplitudes where quadratic effects become nonnegligible.

The set of equations which must be solved (for computing residues of the response) may form a nearly dependent set if the network is a wideband (i.e., its poles are seperated by two or more orders of magnitude) network. Accurate inversion of the corresponding matrix can be quite a formidable task in this case.

In this section, a 12 x 12 matrix generated in the analysis of a wideband network is examined. The equation of interest is

$$R = (A_o)^{-1}Y$$

where R = column vector of residues of poles of the system response

Y = column vector of integrated system outputs at time T

A_o = 12 x 12 matrix whose entries are generated by

$$(A_o)_{ij} = \frac{e^{\lambda_j T}}{\lambda_j^i} - \sum_{m=1}^{i} \left(\frac{T^{m-1}}{(m-1)!} \right) \frac{1}{\lambda_j^{i+1-m}}$$

The linear portion of the system examined has two poles. These poles are

$\lambda_1 = -0.011550998(2\pi)(10^6)$ rad/sec

$\lambda_2 = -10.616986 \quad (2\pi)(10^6)$ rad/sec

The system is excited by the input function

$$X_{in}(t) = [e^{\alpha_1 t} + e^{\alpha_2 t}]\, U(t)$$

where

$\alpha_1 = -10^7$ rad/sec

$\alpha_2 = -1.75 \times 10^7$ rad/sec

The poles of the quadratic response of this network are found in terms of the input and linear transfer function poles as

$$\gamma_1 = \lambda_1$$

$$\gamma_2 = \lambda_2$$

$$\gamma_3 = 2\lambda_1$$

$$\gamma_4 = 2\lambda_2$$

$$\gamma_5 = \lambda_1 + \lambda_2$$

$$\gamma_6 = \alpha_1 + \lambda_1$$

$$\gamma_7 = \alpha_1 + \lambda_2$$

$$\gamma_8 = \alpha_2 + \lambda_1$$

$$\gamma_9 = \alpha_2 + \lambda_2$$

$$\gamma_{10} = 2\alpha_1$$

$$\gamma_{11} = 2\alpha_2$$

$$\gamma_{12} = \alpha_1 + \alpha_2$$

The matrix A_o was generated for T = 600 nanoseconds. The difficulty in inverting this matrix arises because $\gamma_2 \approx \gamma_5$ and consequently

$$C(I,2) \approx C(I,5)$$

for all I, thereby making accurate inversion by standard routines impractical.

The matrix A_o is shown below.

ORIGINAL MATRIX, A_o

```
0.5671237212721573430D+00   0.1503306.0.1257910760D-01   0.3746145570345003300D+00   0.7495304100253353320D-02
0.1497431605403065321D-01   0.3004585065732207700D-02   0.1506557614027274.0D-01   0.5600534056525160550D-01
0.1107523504.0043576D-11   0.4900060273000255300D-01   0.2057342054376411530D-01   0.5636363582815387725D-01

0.1774154201471021590D+00   0.6709646001734112020D-02   0.1740632723015112700D+00   0.4441002880127005110D-02
0.3760359776702075020D-02   0.4 7346551505090050D-01   0.7651.7.5031.52075.0D-02   0.5000500570267255650D-01
0.6908154735430650950D-02   0.2759001530053000300D-01   0.1052655031206350220D-01   0.2043535705365035430D-01

0.3560114720715252245D-01   0.2566347142026345020D-02   0.5522065352441070980D-01   0.1315660072056590650D-02
0.2564105579465634370D-02   0.1203207502562261470D-01   0.2243794357025502750D-02   0.3404400321077310050D-02
0.2054614213706132730D-02   0.7624500231073455550D-02   0.4076540530324603560D-02   0.5600150250678147450D-02

0.5553509320552008140D-02   0.5011632950051552010D-03   0.5307208720184107030D-02   0.2539001167223316030D-03
0.5006783046033027200D-03   0.2290111753453102120D-02   0.4430134771235007300D-03   0.1565021511430754050D-02
0.4031114715635621250D-03   0.3110750033401327100D-02   0.3943043315335027100D-03   0.1000176354105521320D-02

0.6433261.1112230437D-03   0.7343624131702335090D-04   0.5537037062210461630D-03   0.3552610206313458540D-04
0.7336300434365045440D-04   0.5087400241510607250D-03   0.3463017174455340530D-04   0.2131310193703260.1D-03
0.5953352557250000340D-04   0.1990624007903350.0D-03   0.1287154161035016534D-03   0.1564290507559032520D-03

0.6431388990030244360D-04   0.2613060190557030120D-05   0.3400240516051007310D-04   0.4560132200732606.60D-05
0.8604781450161960510D-05   0.5560075262220134350D-04   0.7624637401773030430D-05   0.2445307510543379550D-04
0.6990510614027672500D-05   0.2244667500040331910D-04   0.1403070730.41030480D-04   0.1727527451794134500D-04

0.5520545935324397710D-05   0.8422704013.71201500D-05   0.3434347450504715200D-05   0.4514557162530367550D-06
0.6414040913218320220D-06   0.5003502042287030460D-05   0.7456202603542073790D-06   0.2295574953960774.1D-05
0.6065046031216780260D-06   0.7117556245193034100D-05   0.1427322760731117090D-05   0.1706355055803323.5D-05

0.3823214407074034370D-06   0.7065503559420670700D-07   0.4125543720657401007D-06   0.3024726263730257600D-07
0.7057030306623055070D-07   0.2447023306085704150D-06   0.3260737150330325310D-07   0.1654572732054607220D-06
0.5760123601164533940D-07   0.1710314734542940050D-06   0.1179075121567027020D-06   0.1339240302130039740D-06

0.4703271013003402933D-05   0.5105784102025335750D-06   0.2731203511025506461D-07   0.2635565467430145530D-06
0.5161120350552209454D-03   0.1705309513273000700D-07   0.4613335060532615150D-06   0.1315311066035510540D-07
0.4260541320322052590D-03   0.1223690775583567232D-07   0.6533254735404035.70D-06   0.1005987732275325340D-07

0.6070325052655600400D-05   0.3365725461006703546D-09  -0.2655570340307055600D-09   0.1803012004050611030D-09
0.5302744256657400640D-09   0.1065120523047340420D-08   0.5010..2575637072030D-09   0.3312220797370034000D-09
0.2791999103166010260D-09   0.7757215415553138540D-09   0.5430521000500300790D-09   0.6440560409043246223D-09

0.8376300780050011033D-04   0.1930322779220525119D-10   0.1314030520032407550D-07   0.1100343676111010020D-10
0.1908600142409150340D-10   0.5100191335024742379D-10   0.1773600211045510720D-10   0.4745520220440011030D-10
0.1647205517220637555D-10   0.4447520367535620650D-10   0.3190555101004250720D-10   0.3717197473508037500D-10

-0.1154137100390009640D-12  0.1054100570605112410D-11  -0.8034760456022915340D-07   0.5901245302231023110D-12
0.1003210171572600130D-11   0.3076798446070631410D-11   0.3529605515033064120D-12   0.2170043487321342530D-11
0.8837150250812270600D-12   0.2320505150647770150D-11   0.1605270451710220850D-11   0.1353321545670402730D-11
```

It is readily apparent, through visual inspection, that row scaling is necessary, yielding

```
(ROW) SCALED MATRIX A


 0.14164310488214272D+02    0.36104716047007210300D+00   0.13062527055124904D+02    0.16002350235350051D+00
 0.36125412933495065D+00    0.23094247252179190D+01      0.31450014000140211D+00    0.13723507812002500D+01
 0.28649040601494000D+00    0.12062503391202413270D+01   0.00320531507503070D+00    0.07720981470000750D+00

 0.83892700603631003D+01    0.41460077177820140100D+00   0.02000787440151355D+01    0.20000736781253050D+00
 0.41424250075361391D+00    0.23517540057400470D+01      0.33102001250053336D+00    0.14014120634773323D+01
 0.33025260937535742D+00    0.13003060380330013400D+01   0.77201071400271403D+00    0.30316060350380043D+00

 0.62017212349631984D+01    0.45155621211002002820D+00   0.61375471455310442D+01    0.23140500730520100D+00
 0.45100032670772052D+00    0.22750030017419220300D+01   0.33525530050000330D+00    0.14325731021301211D+01
 0.36144490121700977D+00    0.13413733576420472D+01      0.02206317005100471D+00    0.10203544410778150D+01

 0.51361178454616378D+01    0.48084950713511892D+00      0.50319013673714328D+01    0.24042083555437260D+00
 0.48030506423070078D+00    0.21371300100004404D+01      0.42210725273000000D+00    0.15022938753333544D+01
 0.38875665015407792D+00    0.13511003044803762D+01      0.05006050904071011D+00    0.10536217560041230D+01

 0.44283719400510983D+01    0.50550254749404713D+00      0.43003051363070335D+01    0.26519331403300134D+00
 0.50500522514355800D+00    0.21252845823327997D+01      0.44500190036024345D+00    0.15019302301017000D+01
 0.40846699386410435D+00    0.13702530260375734D+01      0.08001300731102501D+00    0.10767944620023000D+01

 0.39397839713259559D+01    0.52692820290351635D+00      0.30155303734940697D+01    0.27947201788173477D+00
 0.52642172281450560D+00    0.20600140651500024D+01      0.46523510112030307D+00    0.14303527710100841D+01
 0.42766454430817705D+00    0.13732507719033440D+01      0.30707735420300355D+00    0.10935702374043315D+01

 0.35810864245873558D+01    0.54591460083943012D+00      0.35511543312016724D+01    0.29260739855866430D+00
 0.54540174233022870D+00    0.20924359403325600D+01      0.40320773600103533D+00    0.14379252935104128D+01
 0.44435249055650011D+00    0.13725413906431531D+01      0.02523743647512051D+00    0.11059594254928322D+01

 0.30631224739332064D+01    0.56685166356650007D+00      0.33105030005100310D+01    0.30033356922557044D+00
 0.56633071694457942D+00    0.19637321705702103D+01      0.50300050200323072D+00    0.14881306531505134D+01
 0.46389476904668920D+00    0.13730443692200278D+01      0.94320505000110135D+00    0.11226939275420640D+01

 0.41402633950346593D+02    0.45650170740816107D+00      0.24402776650331770D+01    0.24362110089422100D+00
 0.45609115607645906D+00    0.15020521682221055D+01      0.40611170351708740D+00    0.11570615602952507D+01
 0.37594423813019703D+00    0.10772161282100490D+01      0.75117700010052121D+00    0.08556542349315230D+00

-0.62602803740076283D+04    0.34860307921640043D+00     -0.27150031763020442D+00    0.19246457306210080D+00
 0.34834379755018260D+00    0.10847563033537717D+01      0.31100370007646291D+00    0.05660400820405841D+00
 0.23751081865133973D+00    0.79984211321344520D+00      0.55602503156720514D+00    0.06322735402059475D+00

 0.53595115290657197D+05    0.12734609542597463D+00      0.34132434300341740D+02    0.70947833768712133D-01
 0.12723821370094438D+00    0.30314615135230130D+00      0.11330562750774229D+00    0.33381522209593382D+00
 0.10533410663027220D+00    0.20450757502533530D+00      0.20413033305217240D+00    0.23783974510837603D+00

-0.58590331522427290D+03    0.54020049062350765D-01     -0.45357701021403543D+04    0.30364072628010483D-01
 0.53974381734036139D-01    0.15627131850150042D+00      0.47370511200000284D-01    0.12539235902499236D+00
 0.44807223313001543D-01    0.11780157315565205D+00      0.35534030430769467D-01    0.93101330626763471D-01
```

Program Steps Used:

The row scaling indicated on the earlier pages was achieved by choosing (see program description on page 39)

ISP = 1

ISQ = 0

As mentioned earlier, the second and fifth columns of the original matrix A_o form a linearly dependent set. Therefore, a perturbation procedure is needed to obtain the inverse. We used

IPERT = 2

so that the 2,2 entry of the scaled matrix A was perturbed. The amount of perturbation used was FAC = 0.1E-05. The inverse of the perturbed matrix C was obtained by the high precision routine GKRDCT. From C^{-1}, an estimate of A^{-1} was obtained by use of the deperturbation routine DPERT1. This approximate inverse was then refined by use of the routine CORCT2 via the option parameter

ICORCT = 1

The resulting inverse matrix A^{-1} was then descaled. The inverse of the original matrix, A_o^{-1}, as well as the product $A_o^{-1}A_o$ are printed on the succeeding pages.

(Note that a preconditioning of the original matrix was employed by setting ICOND = 1. This caused diagonal entries of A_o to be multiplied by 1.000000001 = 1 + 1.0E-09., INVERSE OF ORIGINAL MATRIX, A_0

```
 0.1137043907039446351D-07  -0.2550425600593407620D-06   0.4570143036210633390D-05  -0.5530517928636451930D-04
-0.1612346035311054450D-03   0.4510947579853370610D-01  -0.1161331210747315643D+01   0.3325300305210340370D+02
-0.4277753110637992580D+05   0.3239445713277060140D+04  -0.1130472502791790420D+05  -0.1003300572040741130D+04

-0.7906237625136143070D+03   0.7026802737023610230D+10  -0.6055526637471090D+11       0.1100254902853060235D+15
-0.1322110013870042700D+14   0.1028806354362032 65D+15  -0.7110657023151042 20D+15   -0.3012597574202502130D+16
 0.3856704203064000330D+17  -0.1590355859211520120D+16  -0.3417516571206470310D+17  -0.1577587112551051200D+16

-0.1557812201431430730D-03   0.3304731026287930040D-02  -0.3 7155355973141027D-01    0.7520039180372001710D+0
 0.2070116101150547740D+01  -0.5737051049120572030D+05   0.2761257310675105600D+05  -0.4431060632544531410D+
 0.5446464615536033201D+07  -0.4136002759967831540D+06   0.1420305725033041 10D+03   0.1053401952000725020D+

-0.6460240071974235550D+07   0.1763361703983311940D+09  -0.2074131756427536001 D+16  0.2940615785532213760D+11
 0.3040057833420903210D+11  -0.7220627130411023630D+13   0.5530460021032370070D+14  -0.3060642353053045020D+15
-0.6728039135133711530D+14   0.5361264644106214000D+16   0.1540530750703053010D+16   0.6063579574679214000D+14

 0.8787027677551504150D+03  -0.3173451824193174540D+10   0.7393305737474704050D+11  -0.1200709427260756030D+13
 0.1321003223636333650D+14  -0.6053367199354177540D+14  -0.5475119377050161070D+15   0.1230038305584116700D+17
-0.1030407026375427360D+16   0.3371013630054032400D+16   0.7701357637217104300D+17   0.3044414301136043510D+16

 0.1615203592833593030D+05  -0.2030501003540015370D+06   0.5313019 [illegible] D+07   -0.1322027 [illegible]
-0.2057172534543074330D+08   0.6050136077205270301D+06   0.2740711 [illegible] D+11   -0.6703071 [illegible]
 0.7758701757305324710D+13  -0.2772976215805712500D+14  -0.5470307303 [illegible] D+15 -0.3207402725333120 70D+1

-0.5902813040451329350D+03   0.1830503910370610 D+10    -0.3467275103540 2140D+11     0.3759300 [illegible] D+1
 0.2682853219530085140D+12  -0.9373771562073113790D+14   0.1 57703034027 310D+16     -0.237 [illegible] D+17
 0.1309751691703767330D+1   -0.3674001500524430420D+16  -0.3 7 6 442 [illegible] D+17  -0.4237567313363014 30D+1

-0.1650021034317016370D+07   0.3330640013405402770D+0   -0.435317307312673 0D+07     0.2417230827 [illegible]
 0.2904272522351721 30D+1    0.4203 0287 [illegible] 715 D+11  -0.5277 35735 2 [illegible] 35D+15  0.15574355 [illegible]
-0.1195401377163521370D+16   0.4255330747011037400D+16   0.5 752 [illegible] D+15   0.4070875052001 [illegible] D+14

 0.5499505430456117230D+0   -0.1013550380740301720D+13   0.2 75070 47235 420D+11     -0.2074 5930 [illegible] 310D+1
-0.2979237533514053460D+12   0.6578065506242053D+14      -0.1717607322 [illegible] 620D+16  0.1255740 [illegible] D+17
-0.7109301350434600300D+17   0.1090600643047373230D+1    0.4 255 31 [illegible] 330D+17  0.2303571100 [illegible] D+16

 0.3738071701664015130D+07  -0.7403002261864327712D+0    0.3 2431271 [illegible] D+0   -0.5631 [illegible]
-0.6546607550573302420D+13  -0.1545355300650 [illegible] D+12  0.3 31 3 7 231 2111D+16  -0.3 [illegible]
 0.3014435904530974075D+16  -0.1072248113660267 D+17    -0.24 537 [illegible] D+16     -0.322 [illegible]

 0.6770509761950873350D+07  -0.14 9 3 [illegible] D+0    [illegible]                  [illegible]
-0.5306060017 5157072220D+1  -0.10 7 7 [illegible]       [illegible]                  [illegible]
 0.9351070311 1017130D+1    -0.24054 [illegible]         [illegible]                  [illegible]

-0.6636337 [illegible]        0.2419 [illegible]          [illegible]                  [illegible]
 0.954 [illegible]            0.7521 [illegible]          [illegible]                  [illegible]
-0.72 9 23311 [illegible]     0.7575 [illegible]          [illegible]                  [illegible]
```

PRODUCT OF ORIGINAL MATRIX, A_0, AND ITS COMPUTED INVERSE

```
 0.1000000020000000100D+01    0.2434466654714.24220D-11    0.2276243.22354.3721D-21   -0.1615070140037665D-21
-0.8240736.3235353.21D-22    -0.55760737.1534173 3D-22     0.1. 035.0603.61745.D-22    -0.3271.3612553027676.-22
-0.9243313224106.4063D-22    -0.2704230063534 41550D-22    0.7463.3.2020723.5220D-21   -0.3277160163565265261-21

-0.2441406230000300000D-33    0.999933331019. 5.3363D+03   0.16  11621132247 .29D-37   -0.6435622024140200170D-37
 0.2344040.34472567350D-37    0.2357260011.. ) ( 31.D-37  -0.1  .0670..07151151179D-36  -0.13...7654.7410140 7D-36
 0.3543181.37457443430D-37    0.5633335422157052257D-37   -0.6125252  47.6444 0D-37    -0.5507260060151663H-36

-0.9334947317729262330D-12    0.1327533252727 22542D-17    0.30000333333333323D+00    0.1282251735305 .. -17
 0.3554031 .6072.7.7.20D-16  -0.6353334352033192020D-17    0.12 .502..725367310D-16    0.3523.672603772.3411-17
 0.3563031125572.00000D-13   -0.3706405263976317110D-17    0.070606551.372467350D-16  -0.4041740350353020.2D-16

 0.0                          0.4003931635006430260D-06    0.16930653.6735.4 290D-07    0.1000060032342526060D+01
 0.3067042.0000412513D-07     0.451713331763261371D-06    -0.6312135.0336060271D-03    0.5211020346.3610.0.0D-06
 0.3391797115077133640D-07    0.73520727207420790.2D-06    0.1.03737427030 .0270D-03    0.5.2202016.727.0310D-06

 0.4882312.000000300000D-03  -0.6323622152667302570D-07   -0.11324.225 752421..D 03    0.1160134615272004250D-06
 0.1000000 .3210370331D+01    0.265560322706209530.D-06    0.1047522227.3.6.1.27D-03    0.1364371920116245750D-06
-0.4193046036.0154523 0D-07   0.351231418037 3703150D-07   -0.6137722245156764.320D-06    0.77535673733342453.0D-07

 0.0                         -0.346490391534045236D-11    -0.3.37972207231712950D-11   -0.3731527 73870722 30D-12
-0.3371927737026102000D-11    0.1000000000000024330.D+01  -0.2.7335501437310132 -11     0.133.70237633565350D-10
-0.1464614 0010 .023720D-11  -0.7387354217277253270D-11    0.1373020001525410350D-11   -0.7310124727760265131-12

-0.4002012530100200300D-05    0.9300031532502026570D-07    0.7152557573360 .7530DD-06    0.3472202273412624 77. --2
 0.9040.627320.3.  .3040D-08 -0.2041277096000 177.72-0     0.19  1339.2711 77112 D+01   -0.5030035.03. 460113D-17
 0.1497437.5173.1377D-07      0.1303332477463433030D-06    0.640361.771.0312 120D- 7     0.1203341337006753.11-07

 0.3314697200625100000D-06   -0.225705133471520034 D-03   -0.7003411249572 3777D-03    -0.1311131130625925.. -0.
-0.1352464.3045033913D-08    -0.5312173211345370219D-03   -0.1253067371 3o1 22217D-03    0.3933332360730772..D+ )
-0.1430475.1377527310 D-08   -0.7314554235509917359D-06   -0.355170464251.0071.D-03    -0.2...1433.5070272140D-06

 0.0                         -0.3732531533770355 09D- 07    0. 43333 45510007773D-06    -0.4 .0 .. 0250 1.022. D-07
-0.8008176596374325430D-07   -0.1573037460534 35 .2D-06   -0.3759.501 40460 .4210D-07   -0.4 .2274 3427030.. 71- 0.
 0.9339390.0600 .324540D+00    0.73024540343507153 0D-07    0.10373354 3214103430D-07    0.17472. 3104425106400 -07

 0.0                          0.4624745031372592 307D-09  -0.3725232 7451914360D-06     -0.7255 133333450410 5D-03
-0.365745.33619.570130D-03   -0.135423232523155745D-0.    -0.11 7552270241 3..1D-0.      0. .... 0102.. 3 3107... -0.
 0.1149306.33033.0122D-06     0.3 33330927714 0133 .5D+ 0  - . 11.042 .5 .0122315170D-06  -0.2..887023353.3154.311D-06

-0.457763071875 330000D-04   -0.3603350243511132210D-1.    0.. 3202572307730..1259D-07     0.2282572542120271 1-06
 0.2620345.7 7135.7203D-10    0.2553074721311.6 0737D-07  -0. ..50 30.351 75352110D-1.    -0.35..3235 34722 75..3 -0.
 0.. 30.030 -25252 .0730 -03  0.2511311067345311 6 20D-06    0.1933 133 0207793 .. +00    -0.  35410.0011530734. -06

 0.610351  230030000.00 -04  -0.5125335632920017.20D-03   -0.2 .770 .2  925333.40D-17     0.1.31.334.2020 0202D-06
 0.1065603.3045 723472D-10    0.25 .1574334207300.7 D-08  -0.131479 056.  3413 D-1.     -0..  33333441253345 2. -06
-0.1004 2  .220533 570 -10    0.5 . 311.553255227 60 D-08   0.  133 0  0727 11  D-03     0.1033303331006 43. .D+01
```

(MAT-I): RMSE= 0.6137D-04   MAX= 0.4823D-03
```
*******************************************************************************************************
```

3.5 Program Description

This FORTRAN IV program performs high-precision matrix inversion. This computer program has capabilities for automatic adaptive scaling of the original matrix,application of perturbation techniques in finding the inverse and iterative correction of an (approximate) inverse matrix; each of these has been discussed in the earlier sections.

To enable the test engineer to effectively use the program, a description of the input data cards is given below.

Input Data Deck

The input data deck consists of one card containing input variables, followed by $[\frac{N^2+2}{3}]$ cards containing the entries of the matrix to be inverted, where N is the dimension of the matrix and $[X]$ is the truncation function.

CARD #1 Option card which contains eight variables.

Variable Name (Format)	Description	Columns
N(I2)	dimension of the square matrix to be inverted	1-2
ISP(I2)	adaptive scaling option ISP = 0 row scaling is not performed ISP = 1 row scaling is performed	3-4
ISQ(I2)	adaptive scaling option ISQ = 0 column scaling is not performed ISQ = 1 column scaling is performed	5-6
IPERT(I2)	diagonal perturbation option IPERT = 0 perturbation is not employed IPERT = 1,2,...,N.perturbation of the $(A_o)_{IPERT,IPERT}$ entry is performed IPERT = N + 1 perturbation of the entire diagonal is carried out	7-8
ISLID(I2)	de-perturbation option to be used with IPERT = N+1 ISLID= 0 deperturbation is performed with subroutine DPERT2 ISLID= 1 deperturbation is performed through a "sliding' correction method (utilizing a family of matrices)	9-10

Variable Name (Format)	Description	Columns
ICOND(I2)	Pre-conditioning option ICOND = 0 no conditioning of the original matrix is employed ICOND = 1 the original matrix A_o is preconditioned with a multiplier 1 + 1.0 E-9 along the diagonal	11-12
ICORCT(I2)	option to be used in conjunction with IPERT = 0 ICORCT = 0 iterative correction is not used when IPERT = 0 ICORCT = 1 iterative correction is used when IPERT = 0	13-14
IPRINT(I2)	printing option IPRINT = 0 suppresses printing of many of the intermediate matrix quantities used for computation IPRINT = 1 all intermediate matrices are printed	15-16
CARD #2 through	card #[$\frac{N^2}{3}$ + 1] These cards contain the matrix values, and are read in 3D25.18 format. The matrix entries are entered on the cards in an order prescribed by columns. That is, they are entered as first row, columns 1 through N; second row, columns 1 through N; etc.	
END OF FILE CARD		

Note a logical flow diagram depicting the effect of various option parameters is given below.

A listing of the complete FORTRAN program is given in Appendix B.

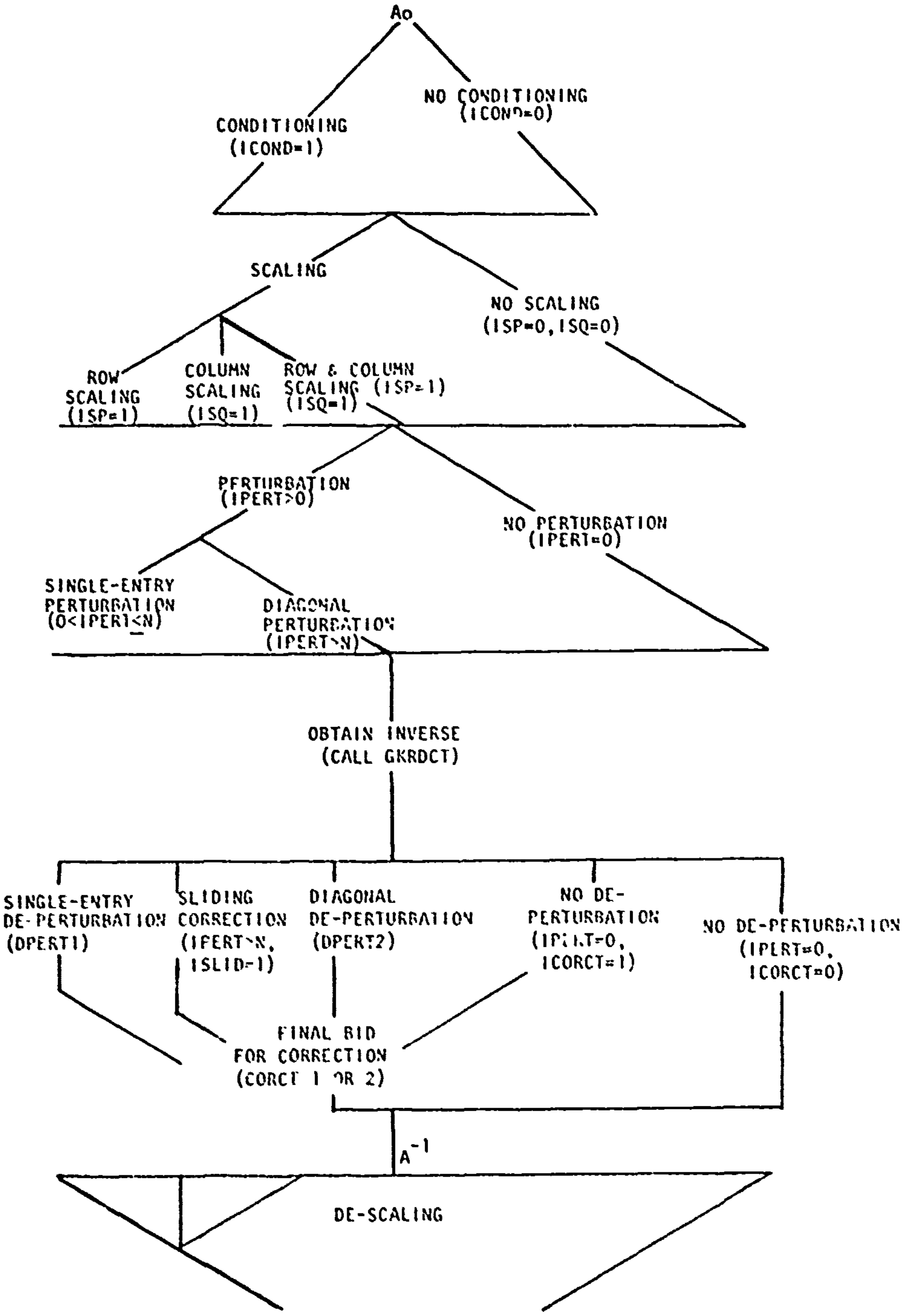

Ao
CONDITIONING (ICOND=1)
NO CONDITIONING (ICOND=0)
SCALING
NO SCALING (ISP=0,ISQ=0)
ROW SCALING (ISP=1)
COLUMN SCALING (ISQ=1)
ROW & COLUMN SCALING (ISP=1) (ISQ=1)
PERTURBATION (IPERT>0)
NO PERTURBATION (IPERT=0)
SINGLE-ENTRY PERTURBATION (0<IPERT≤N)
DIAGONAL PERTURBATION (IPERT>N)
OBTAIN INVERSE (CALL GKRDCT)
SINGLE-ENTRY DE-PERTURBATION (DPERT1)
SLIDING CORRECTION (IPERT>N, ISLID=1)
DIAGONAL DE-PERTURBATION (DPERT2)
NO DE-PERTURBATION (IPERT=0, ICORCT=1)
NO DE-PERTURBATION (IPERT=0, ICORCT=0)
FINAL BID FOR CORRECTION (CORCT 1 OR 2)
A^-1
DE-SCALING

Fig. 3 Flow diagram of HPMINV program options

REFERENCES

[1] E. J. Ewen and D. D. Weiner, "Identification of Weakly Nonlinear Systems," Midwest Symposium on Circuits and Systems (Lubbock, Texas), Aug. 1977.

[2] E.Ewen, "Black-Box Identification of Nonlinear Volterra Systems," Ph.D. Dissertation, Syracuse University, Dec. 1975.

[3] J. Bussgang, L. Ehrman, and J. Graham, "Analysis of Nonlinear Systems with Multiple Inputs," Proc. IEEE, Vol.62, pp 1088-1119, Aug. 1974.

[4] S. Narayanan, "Application of Volterra Series to Intermodulation Distortion Analysis of Transistor Feedback Amplifiers," IEEE Trans. Circuit Theory, Vol. CT-17, pp 518-527, Nov. 1970.

[5] H. J. Kuno, "Analysis of Nonlinear Characteristics and Transient Response of IMPATT Amplifiers," IEEE Trans. Microwave Theory Tech., Vol. MTT-21, pp 694-702, Nov. 1973.

[6] V. K. Jain, "Filter Analysis by use of pencil of functions," IEEE Rans. Circuits and Systems, Vol. CAS-21, pp 580-583, Sept. 1974.

[7] V. K. Jain, D. D. Weiner, J. Nebat and T. K. Sarkar, "System identification by pencil of functions method," Proc. RADC Workshop on Spectral Analysis, pp 99-102, May 1978.

[8] M. L. Van Blaricum and R. Mittra, "A technique for extracting the poles and residues of a system directly from its transient response," IEEE Trans. Ant. Prop., AP-23, pp 777-781, 1975.

[9] K. J. Iliff and L. W. Taylor, "Determination of stability derivatives from data using a Newton-Raphson minimization technique," NASA technical report, TN D-6579, 1972.

[10] J. A. Cadzow, "Recursive digital filter synthesis via gradient based algorithms," IEEE Trans. Acous. Sp. Signal Proc., Vol ASSP-24, pp 349-355, Oct. 1976.

[11] K. J. Astrom and P. Eykhoff, "System identification - A survey," Automatica, pp. 123-162, 1971.

[12] E. G. Evans and Fischl, "Optimal least-squares time-domain synthesis of resursive digital filters, "IEEE Trans. Audio Electroacous., Vol. AU-21, pp. 61-65, Feb. 1973.

[13] M. J. Levin, "Estimation of asystem pulse transfer function in the presence of noise," IEEE Trans. Autom. Control, Vol. AC-9, pp 229-235, July 1964.

[14] R. L. Kashyap, "Maximum likelihood identification of stochastic linear systems," IEEE Trans. Autom. Control, Vol. AC-15, pp. 25-34, Feb. 1970.

[15] R. K. Mehra, "Identification of stochastic linear dynamic systems using Kalman filter representation," AIAA J., Vol. 9, pp. 28-31, Jan. 1971.

[16] E. W. Chenny, Introduction to Approximation Theory. New York: McGraw-Hill 1966.

[17] W. D. Stanley, Digital Signal Processing. Reston (Prentice-Hall): Reston, 1975.

[18] D. K. Faddeev and V. N. Faddeeva, Computational Methods of Linear Algebra. San Francisco: W. H. Freeman and Co., 1963.

[19] G. H. Stewart, Introduction to Matrix Computations. New York: Academic Press, 1973.

[20] J. H. Wilkinson, The Algebraic Eigenvalue Problem. Oxford: Clarendon Press, 1965.

APPENDIX A

IGRAM PROGRAM
LISTING

```
C      PROGRAM NAME: IGRAM
C      NETWORK TRANSFER-FUNCTION IDENTIFICATION ROUTINE UTILIZING
C      PENCIL-OF-FUNCTIONS METHOD (PURE INTEGRATORS USED)
C
       DIMENSION X(1024),V(1024),XORG(1024),VORG(1024),XREC(1024)
       DIMENSION DATA(1024,2),DATA2(1024,2),BUFF(3072)
       DIMENSION G(20,20),Z(20,20),GAMMA(20),XLAMDA(20),COEFF(20)
       DIMENSION NPULSE(64)
       DIMENSION TITLE(20)
       REAL*8 G,Z,GAMMA,XLAMDA,COEFF
       REAL*8 DELTA,Q,QSAV,DELSAV,AVGQ,AVGN,SUMV2,XSAV
       EQUIVALENCE (Z(1,1),BUFF(1)),(G(1,1),BUFF(1501))
       EQUIVALENCE (DATA(1,1),X(1)),(DATA(1,2),V(1))
       EQUIVALENCE (XORG(1),DATA2(1,1)),(XREC(1),DATA2(1,2))
       COMMON /GKRD/IGKR
       COMMON /NUMER/NN
       COMMON /BIAS/IBIAS
C
       RDEL=0.01
       MAX=20
       MAXPL=1024
       NPT=MAXPL
C
       WRITE(6,2)
       WRITE(6,1022)
       READ(5,1021)TITLE
       WRITE(6,1021)TITLE
       WRITE(6,1023)
4320   READ(5,1001)N,MP1,IPLT
       READ(5,6995)(XORG(K),K=1,MP1),(VORG(K),K=1,MP1)
4321   READ(5,1,END=1234)N,MP1,ISKIP,IREM,IBIAS,DELTA
       IF(N.EQ.10000)GO TO 4320
       WRITE(6,1000)N,MP1,DELTA,IREM,IBIAS
C
       IGKR=1
       IZTS=1
       QSAV=1.0D00
       Q=QSAV
       NM1=N-1
       NP1=N+1
       NP2=N+2
       NPNP1=N+N+1
       NPNP2=N+N+2
       NN=N-IREM
C
C      GENERATING SEQUENCE X(K)
23     DO24I=1,NP1
```

```
      V(I)=VORG(I)
24    X(I)=XORG(I)
      IF(IPLT.EQ.0)GO TO 6633
      WRITE(6,1003)
      CALL PLOTIT(DATA ,2,MP1,1,MP1,ISKIP,MAXPL,1,1.0)
6633  CONTINUE
C
C     START IDENTIFICATION FROM INPUT OUTPUT DATA
C
      CALL GRAMI(X,V,MP1,N,DELTA,Q,IZTS,GAMMA,XLAMDA,G,Z,MAX,IREM)
      CALL ERROR(XREC,VORG,GAMMA,MP1,N,XLAMDA,XORG)
C     PLOT RECONSTRUCTION
      WRITE(6,66)
      WRITE(6,1004)
      IF(IPLT.EQ.0) GO TO 6544
      CALL PLOTIT(DATA2,2,MP1,1,MP1,ISKIP,MAXPL,1,1.0)
6544  CONTINUE
C
      WRITE(6,2)
      WRITE(6,1022)
100   GO TO 4321
1234  CONTINUE
C
C
1     FORMAT(5I5,3F5.0)
2        FORMAT('1')
66    FORMAT(//,1X,'TRUE RESPONSE VERSES RECONSTRUCTED RESPONSE',//)
1000  FORMAT(1H1,50X,'STARTING SIMULATION',/20X,'SYSTEM ORDER = ',I5,
     1/,20X,'M + 1 = ',I5,///,20X,'SAMPLING INTERVAL = ',F10.6,//,20X,
     2'IREM = ',I5,/,20X,'IBIAS = ',I5,/)
1001  FORMAT(8I5)
1003  FORMAT(//,5X,'INPUT (+) AND OUTPUT (*) OF THE PLANT',/)
1004  FORMAT(//,5X,' OUTPUT (*) AND RECONSTRUCTION (+) ',/)
1021  FORMAT(80A1)
1022  FORMAT(///////////////////////////////////////////////////////////////)
1023  FORMAT(/,1X,'***************************************************',
     1 '************************************')
6995  FORMAT(8F10.0)
      STOP
C
C        DEFINITION OF PARAMETERS USED IN THE SIMULATION OF A
C        LINEAR DYNAMIC SYSTEM
C
C        X IS THE CORRUPTED OUTPUT SEQUENCE
C        V IS THE CORRUPTED INPUT SEQUENCE
C        GAMMA IS THE COEFFICIENT VECTOR
C
C        MAX = ACTUAL DIMENSION SIZE OF 2-DIM ARRAYS IN THE DIMENSION
C        STATEMENT
C        N = ORDER OF SYSTEM
C        THE MAXIMUM VALUE OF N IS MAX/2-1
C        MP1 = M+1, THE TOTAL NUMBER OF SAMPLED POINTS IN EACH SEQUENCE
C
C        RHO = EXPECTATION( W(K)*Q(K) )
C
C        DELTA IS THE SAMPLING INTERVAL
C
C        IGRM = 1 GRAMI IS PERFORMED
C
C        IPLT=0  NO  PLOTS
C        IPLT=1  PLOTS ONLY WITH PRINTER
C
C        IBIAS =0 NO BIAS IS ASSUMED PRESENT ON INPUT-OUTPUT DATA
C        IBIAS =1 SMALL VALUES OF INPUT-OUTPUT BIAS ARE ASSUMED PRESENT
C                 ON THE DATA.
C
       END
      SUBROUTINE GRAMI(X,V,MP1,N,DELTA,QSAV,KOPT,GAMMA,XLAMDA,G,Z,MAX,
     1IREM)
C     THIS SUBROUTINE PERFORMS THE GRAMI TECHNIQUE
C
      DIMENSION X(1),V(1),G(MAX,1),Z(MAX,1),GAMMA(1),XLAMDA(1),Q(20),
     1DEL(20)
      DIMENSION GAM(25)
      DOUBLE PRECISION GAM
      DOUBLE PRECISION G,Z,GAMMA,XLAMDA,DELTA,DEL,PROD,Q,QSAV
      REAL*8 VARQ,VARW,FAC
      REAL*8 SCALE(20),SCAL
      COMMON /GKRD/IGKR
      COMMON /NUMER/NN
      COMMON /BIAS/IBIAS
C
      WRITE(6,1000)
1000  FORMAT(1H1,20X,'THE GRAM I TECHNIQUE')
C        JOPT = 0 IF DIRECT TRANSMISSION IS ASSUMMED
      JOPT=0
      IDLY=IREM
      IF(IREM.NE.0)JOPT=1
C
C        DEL IS THE NUMERATOR OF THE KNOWN FIRST ORDER DIGITAL FILTERS
      DO19I=1,N
      DEL(I)=1.0D00
19    Q(I)=QSAV
      WRITE(6,2020)
2020  FORMAT(30X,'Q PARAMETERS')
      CALL PRVEC(Q,N)
      NP1=N+1
      NP2=N+2
      NPNP1=N+N+1
      NPNP2=N+N+2
      NR=MP1-IREM
      NP1PIR=NP1+IREM
      DO 12 I=1,MAX
      DO 12 J=1,MAX
12    Z(I,J)=0.0D00
      VARQ=0.0
      VARW=0.0
      DO 300 I=1,MP1
      VARW=VARW+V(I)*V(I)
300   VARQ=VARQ+X(I)*X(I)
      VARQ=DSQRT(VARQ/MP1)
      VARW=DSQRT(VARW/MP1)
C
C
C        CALCULATING THE G MATRIX
      IF(IBIAS.EQ.0)GO TO 11
      NPNP2=N+N+2+1
      NR=MP1-IREM+1
11    CONTINUE
C
      DO10I=1,NPNP2
      GAM(I)=0.0
      GAMMA(I)=0.0D00
      DO10J=1,NPNP2
10    G(I,J)=0.0D00
      GAM(1)=1.0
      DO50K=1,MP1
      IF(K-IDLY)25,25,24
25    GAMMA(NP2)=0.0D00
      GO TO 26
24    FAC=1.0
      GAMMA(NP2)=V(K-IDLY)/FAC
      FAC=1.0
      GAMMA(1)=X(K)/FAC
26    CONTINUE
      DO30I=1,N
      GAM(I+1)=GAM(I+1)*Q(I)+GAM(I)*DEL(I)
      GAMMA(I+1)=GAMMA(I)*DEL(I)+GAMMA(I+1)*Q(I)
30    GAMMA(I+NP2)=GAMMA(I+NP1)*DEL(I)+GAMMA(I+NP2)*Q(I)
         IF(IBIAS.EQ.1)GAMMA(NPNP2)=GAM(I+1)
      DO 40 I=1,NPNP2
           DO 40 J=I,NPNP2
40    G(I,J)=G(I,J)+GAMMA(I)*GAMMA(J)
50    CONTINUE
C     PREFORM SCALING ON G-MATRIX
      DO 735 I=1,NPNP2
735   SCALE(I)=DSQRT(G(I,I))
      DO 736 I=1,NPNP2
      DO 736 J=1,NPNP2
736   G(I,J)=G(I,J)/(SCALE(I)*SCALE(J))
      WRITE(6,1009)
1009  FORMAT(10X, ---G MATRIX---')
      DO 55 I=1,NPNP2
55    WRITE(6,3)(G(J,I),J=1,I)
3     FORMAT(1X,10D13.5)
42       CONTINUE
      DO60I=2,NPNP2
      K=I-1
      DO60J=1,K
60    G(I,J)=G(J,I)
C
      IF(JOPT)70,90,70
70    DO80J=1,NPNP2
      DO80I=1,NR
      G(NP1+I,J)=G(NP1PIR+I,J)
80    CONTINUE
      NPNP2=NPNP2-IREM
      DO85J=1,NPNP2
       SCALE(NP1+J)=SCALE(NP1PIR+J)
      DO85I=1,NR
      G(J,NP1+I)=G(J,NP1PIR+I)
85    CONTINUE
90    CONTINUE
      CALL GKRDCT(G,Z,XLAMDA,N,NPNP2,MAX)
C     DE-SCALE SYNTHETIC COEFFICIENT VECTOR, XLAMDA
      DO 741 I=1,NPNP2
741   XLAMDA(I)=XLAMDA(I)/SCALE(I)
      IF(IBIAS.EQ.0)GO TO 43
      NPNP2=NPNP2-1
      NR=NR-1
43    CONTINUE
      XMEAN=XLAMDA(NPNP2+1)
      IF(JOPT)120,130,120
120   NPNP2=NPNP2+IREM
      DO122I=1,NR
122   XLAMDA(NPNP2-I+1)=XLAMDA(NP2+NR-I)
      DO123I=1,IREM
123   XLAMDA(NP1+I)=0.0D00
130   CONTINUE
      FAC=1.0
      DO301I=NP2,NPNP2
301   XLAMDA(I)=XLAMDA(I)*FAC
      WRITE(6,1001)
1001  FORMAT(10X,'THE SYNTHETIC COEFFICIENT VECTOR, XLAMDA, IS')
      CALL PRVEC(XLAMDA,NPNP2)
      DO 150 I=1,NPNP2
150   GAMMA(I)=XLAMDA(I)
C
C         GENERATING GAMMA FROM XLAMDA
```

```
      CALL BUIL [illegible] DEL,N,MAX)
      DO160I=1,N[illegible]
      GAMMA(I)=0.0D00
      DO160J=1,NPNP2
160   GAMMA(I)=GAMMA(I)+G(I,J)*XLAMDA(J)
165   CONTINUE
      DO200I=2,NPNP2
200   GAMMA(I)=GAMMA(I)/GAMMA(1)
      GAMMA(1)=1.0D00
      IF(IDLY.EQ.0)GO TO 172
      IDLY1=IDLY+1
      DO 170 II=IDLY1,NP1
      I=NPNP2+1-II
170   GAMMA(I+IDLY)=GAMMA(I)
      DO 172 I=1,IDLY
      GAMMA(I+NP1)=0.0D00
172   CONTINUE
C
C        CALCULATING THE EQUIVALENT CONTINUOUS DESCRIPTION
      CALL IZTOS(GAMMA,N,DELTA,KOPT)
      WRITE(6,1003)
1003  FORMAT(///,1X,100(1H-),/,1X,100(1H-))
      RETURN
      END
```

```
      SUBROUTINE GKRDCT(X,Y,XLAMDA,NN,N,MAX)
      REAL*8 X(MAX,1),Y(MAX,1),A,B,C,D,E,DET,CCC(20,20),XLAMDA(1)
      INTEGER NUM(2,20)
      COMMON /GKRD/IGKR
C     IGKR=0  USE IS MADE OF THE FIRST ROW OF ADJOINT
C          1   DIAGONAL(NEGATIVE ENTRIES SET TO ZERO)
C          2   ABSOLUTE VALUE OF DIAGONAL
      DO 6 I=1,N
      DO 6 J=1,N
6     Y(J,I)=X(J,I)
      A=1.0D0
      DO 43 I=1,N
      B=0.0D0
      L=I
      M=I
C
C        FIND LARGEST ENTRY A(L,M) IN LOWER DIAGONAL SUBMATRIX
C
      DO 18 J=I,N
      DO 18 K=I,N
      IF(DABS(Y(K,J)).LE.B)GO TO 18
      B=DABS(Y(K,J))
      L=K
      M=J
18    CONTINUE
C
C        INTERCHANGE ROWS
C
      IF(L.EQ.I)GO TO 24
      DO 23 J=1,N
      C=Y(L,J)
      Y(L,J)=Y(I,J)
```

```
23    Y(I,J)=C
C
C        INTERCHANGE COLUMNS
C
24    IF(M.EQ.I)GO TO 29
      DO 28 J=1,N
      C=Y(J,M)
      Y(J,M)=Y(J,I)
28    Y(J,I)=C
C
C        BEGIN SWEEP COLUMNS TO THE RIGHT
C        ARRAYS NUM(1,.) ,NUM(2,.) KEEP RECORD
C        OF ROW AND COLUMN INTERCHANGES
C
29    NUM(1,I)=L
      NUM(2,I)=M
      B=Y(I,I)
      Y(I,I)=A
      DO 42 J=1,N
      IF(J.EQ.I)GO TO 42
      C=-Y(I,J)
      Y(I,J)=0.0D0
      DO 41 K=1,N
      D=Y(K,I)*C
      E=Y(K,J)*B+D
C     IF(DABS(E).LT.1.0D-10*DABS(D))E=0.0D0
41    Y(K,J)=E/A
42    CONTINUE
43    A=B
C
C        RESTORE COLUMNS
C
      DO 58 I=2,N
      J=N+1-I
      K=NUM(2,J)
      IF(K.EQ.J)GO TO 52
      DO 51 L=1,N
      C=Y(K,L)
      Y(K,L)=Y(J,L)
51    Y(J,L)=C
52    K=NUM(1,J)
C
C        RESTORE ROWS
C
      IF(K.EQ.J)GO TO 58
      DO 57 L=1,N
      C=Y(L,K)
      Y(L,K)=Y(L,J)
57    Y(L,J)=C
58    CONTINUE
      DET=A
C     *********  SET IPRINT *****
      IPRINT=0
      IPRINT=1
      IF(IPRINT.NE.1)GO TO 111
      WRITE(6,101)
101   FORMAT(/1X,'DET OF GRAM MATRIX IS ')
      CALL PRVEC(DET,1)
      WRITE(6,102)
102   FORMAT(/1X,'ADJOINT MATRIX IS')
      CALL PRMAT(Y,N,N,MAX)
      DO100I=1,N
      DO100J=1,N
      CCC(I,J)=0.0D00
      DO100K=1,N
100   CCC(I,J)=CCC(I,J)+X(I,K)*Y(K,J)
```

```
      WRITE(6,103)
103   FORMAT(/1X,'PRODUCT OF ADJOINT AND GRAM MATRICES')
      CALL PRMAT(CCC,N,N,20)
111   CONTINUE
      DO200I=2,N
      IF(IGKR.EQ.0)XLAMDA(I)=Y(I,1)/Y(1,1)
      IF(IGKR.EQ.0)GO TO 200
      A=Y(I,I)
      IF(Y(I,I).LT.0.)A=0.0D00
      IF(IGKR.EQ.2)A=DABS(Y(I,I))
      XLAMDA(I)=DSQRT(A/Y(1,1))
      IF(Y(I,1).LT.0.0)XLAMDA(I)=-XLAMDA(I)
200   CONTINUE
      XLAMDA(1)=1.0D0
      RETURN
      END
```

```
      SUBROUTINE IZTOS(GAMMA,N,DELTA,IZTS)
C
C        IZTOS SEPARATES THE NUMERATOR FROM THE DENOMINATOR PARAMETERS
C        IN GAMMA
C
      DIMENSION GAMMA(1),X1(10),X2(10)
      DOUBLE PRECISION GAMMA,X1,X2,DELTA
      COMMON /NUMER/NN
      NP1=N+1
200   DO3I=1,NP1
      X1(I)=GAMMA(I)
3     X2(I)=-GAMMA(NP1+I)
      CALL ZTOS(X1,X2,N,DELTA,IZTS)
      IZTS=IZTS+1
      IF(IZTS.EQ.4) GO TO 200
      RETURN
      END
      SUBROUTINE ZTOS(B,A,N,DELTA,IZTS)
      COMMON /NUMER/NN
C
C
C     CONVERSION OF A DISCRETE TIME SYSTEM H(Z) TO A CONTINUOUS TIME SYSTEM H(S)
C
C
C     H(Z)=(A(1) + A(2)*ZETA +....)/(1 + B(2)*ZETA +....)
C                       ZETA = 1/Z
C
C     H(S)=(A(1) + A(2)*S + ....... + A(N+1)*S**N)/DENOM
C
C             DENOM=B(1) + B(2)*S + ..... + B(N+1)*S**N
C
C                  B(1) = 1 ALWAYS
C
C
       DIMENSION B(9),A(9),TEMP(20),RR(20),RI(20),CR(20),CA(20),CAA(20),
      1CA1(20),CB(20),CF(20),CF1(20),CG(20)
       COMPLEX*16 CA,CAA,CA1,CB,CR,CON1,CON2,CONT,FAC,A1,A2,B1,B2,AA1,BB1
      1CG,CF1,CF
       REAL*8 B,A,TEMP,RR,RI,DELTA
       CONT=0.0D00
       IORP=IZTS
       NP1=N+1
       NNP1=NN+1
       A1=0.0D0
       B1=0.0D0
       DO 30 I=1,NP1
       IF(I.LE.NNP1)A1=A1+A(I)
 30    B1=B1+B(I)
       WRITE(6,989) NN
 989   FORMAT(10X,'NN = ',I5)
 999   FORMAT(///)
       WRITE(6,999)
       WRITE(6,1000)
 1000  FORMAT(' Z-DOMAIN DENOMINATOR')
       CALL PRVEC(B,NP1)
       WRITE(6,1001)
 1001  FORMAT(' Z-DOMAIN NUMERATOR')
       CALL PRVEC(A,NP1)
       IF(IZTS.EQ.0) GO TO 310
       IF(IZTS.EQ.1) GO TO 200
       IF(IZTS.EQ.2) GO TO 250
       IF(IZTS.EQ.3) GO TO 200
       IF(IZTS.EQ.4) GO TO 250
 200   CONTINUE
C
C
C      LOGARITHMIC TRANSFORMATION
C
C
C      WORK ON NUMERATOR
C
C
       IF(NN.EQ.0)GO TO 469
       CALL POLRT(A,TEMP,NN,RR,RI,IER)
       DO15 I=1,NN
 15    CA(I)=DCMPLX(RR(I),RI(I))
       DO 7 I=1,NN
 7     CA(I)=(+1.0/DELTA)*CDLOG(CA(I))
       IF(NN.EQ.N) GOTO471
 469   CONTINUE
       DO 470 I=NNP1,NP1
       CAA(I)=0.0D0
 470   CA(I)=0.0D0
```

```
471   CONTINUE
      IF(NN.EQ.0)CAA(1)=1.0D0
C
C
C     NOW THE FIRST NN ENTRIES OF CA CONTAIN THE S-DOMAIN ZEROES OF NUMERATOR
C     AND THE REAMINING ENTRIES ARE ZEROED OUT.
C
      IF(NN.NE.0)CALL POLCON(CA,CAA,0,N)
C
C
C     WORK ON DENOMINATOR
C
C
919   CALL POLRT(B,TEMP,N,RR,RI,IER)
      DO16 I=1,N
      CR(I)=DCMPLX(RR(I),RI(I))
16    CF(I)=1.0D00/CR(I)
909   WRITE(6,1002)
      CALL PRCVEC(CF,N)
      IF(IZTS.EQ.0) GO TO 900
235   DO6 I=1,N
6     CR(I)=(-1.0/DELTA)*CDLOG(CR(I))
      WRITE(6,240)
240   FORMAT(' LOGARITHMIC TRANSFORMATION')
      WRITE(6,999)
      WRITE(6,2000)
2000  FORMAT(' POLES IN S DOMAIN')
      CALL PRCVEC(CR,N)
      DO3000I=1,N
3000  CR(I)=-CR(I)
      CALL POLCON(CR,CB,0,N)
C
C
C     ADJUST DC GAIN CONSTANT
C
C
      A2=CAA(1)
      B2=CB(1)
      FAC=(A1/B1)*(B2/A2)
      DO 603 I=1,NNP1
603   CAA(I)=CAA(I)*FAC
      GO TO 2010
C
C
C     DELAYED PULSE INVARIANT TRANSFORMATION
C
C
C     SHIFTS NUMERATOR COEFFICIENTS FOR DELAY
C
C
250   CONT=A(1)
      DO 300 I=1,N
300   A(I)=A(I+1)-CONT*B(I+1)
      A(NP1)=0.0
400   CALL POLRT(B,TEMP,N,RR,RI,IER)
      DO61I=1,N
      CR(I)=DCMPLX(RR(I),RI(I))
61    CF(I)=1.0D00/CR(I)
      WRITE(6,1002)
1002  FORMAT(1X,'THE POLES OF THE Z-DOMAIN')
      CALL PRCVEC(CF,N)
C
C
C     PARTIAL FRACTION EXPANSION
C
C
```

```
      DO3I=1,N
      CON1=1.0D00
      CON2=0.0D00
      DO4J=1,N
      CON2=CON2*CR(I)+A(N-J+1)
      IF(I-J)5,4,5
5     CON1=CON1*(1.0D00-CR(I)*CF(J))
4     CONTINUE
3     CA(I)=CON2/CON1
C
C
C     TRANSFORMATION OF DENOMINATOR AND NUMERATOR
C
C
224   DO2I=1,N
      CR(I)=CDLOG(CR(I))/DELTA
      CA(I)=CA(I)*CR(I)/(1.0D00-CF(I))
2     CONTINUE
      CA(NP1)=0.0D00
      WRITE(6,241)
241   FORMAT(' DELAYED PULSE TRANSFORMATION')
      WRITE(6,999)
226   WRITE(6,1004)
1004  FORMAT(' NEGATIVE OF THE POLES IN THE S-DOMAIN')
      CALL PRCVEC(CR,N)
      WRITE(6,1003)
1003  FORMAT(1X,'NUMERATOR CONSTANTS OF FACTORIZED H(S)')
      CALL PRCVEC(CA,N)
      CALL POLCON(CR,CB,0,N)
      DO71I=1,NP1
71    CAA(I)=0.0D00
      DO9K=1,N
      CALL POLCON(CR,CF1,K,N)
      DO9J=1,N
9     CAA(J)=CAA(J)+CF1(J)*CA(K)
      CAA(NP1)=0.0D00
2010  CONTINUE
      DO 450 I=1,NP1
450   CAA(I)=CAA(I)+CONT*CB(I)
C
C
C
403   WRITE(6,1005)
1005  FORMAT(' S-DOMAIN DENOMINATOR')
      CALL PRCVEC(CB,NP1)
      WRITE(6,1006)
1006  FORMAT(' S-DOMAIN NUMERATOR')
      CALL PRCVEC(CAA,NP1)
      DO20I=1,NP1
      B(I)=CB(I)
20    A(I)=CAA(I)
900   RETURN
      END
      SUBROUTINE POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
C
C           COMPUTES THE REAL AND COMPLEX ROOTS OF A REAL POLYNOMIAL
C
C        DESCRIPTION OF PARAMETERS
C           XCOF -VECTOR OF M+1 COEFFICIENTS OF THE POLYNOMIAL
C                 ORDERED FROM SMALLEST TO LARGEST POWER
C           COF  -WORKING VECTOR OF LENGTH M+1
C           M    -ORDER OF POLYNOMIAL
C           ROOTR-RESULTANT VECTOR OF LENGTH M CONTAINING REAL ROOTS
C                 OF THE POLYNOMIAL
C           ROOTI-RESULTANT VECTOR OF LENGTH M CONTAINING THE
C                 CORRESPONDING IMAGINARY ROOTS OF THE POLYNOMIAL
C           IER  -ERROR CODE WHERE
C                 IER=0  NO ERROR
C                 IER=1  M LESS THAN ONE
C                 IER=2  M GREATER THAN 36
C                 IER=3  UNABLE TO DETERMINE ROOT WITH 500 INTERATIONS
C                        ON 5 STARTING VALUES
C                 IER=4  HIGH ORDER COEFFICIENT IS ZERO
C
      DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)
      DOUBLE PRECISION XO,YO,X,Y,XPR,YPR,UX,UY,V,YT,XT,U,XT2,YT2,SUMSQ,
     1 DX,DY,TEMP,ALPHA,XCOF,COF,ROOTR,ROOTI,ER1,ER2,XSS,XS,YSS,YS,TOL
C
C           LIMITED TO 36TH ORDER POLYNOMIAL OR LESS.
C           FLOATING POINT OVERFLOW MAY OCCUR FOR HIGH ORDER
C           POLYNOMIALS BUT WILL NOT AFFECT THE ACCURACY OF THE RESULTS.
C
C        METHOD
C           NEWTON-RAPHSON ITERATIVE TECHNIQUE.  THE FINAL ITERATIONS
C           ON EACH ROOT ARE PERFORMED USING THE ORIGINAL POLYNOMIAL
C           RATHER THAN THE REDUCED POLYNOMIAL TO AVOID ACCUMULATED
C           ERRORS IN THE REDUCED POLYNOMIAL.
C
      ER2=1.0D+50
      TOL=1.0D-8
      IFIT=0
      N=M
      IER=0
      IF(XCOF(N+1))10,25,10
   10 IF(N) 15,15,32
C
C        SET ERROR CODE TO 1
C
   15 IER=1
   20 IF(IER)200,201,200
200   WRITE(6,203)IER
203   FORMAT(1X,'ERROR CALLED FROM POLRT, IER = ',I3)
201   RETURN
C
C        SET ERROR CODE TO 4
C
   25 IER=4
      GO TO 20
C
C        SET ERROR CODE TO 2
C
   30 IER=2
      GO TO 20
   32 IF(N-36) 35,35,30
   35 NX=N
      NXX=N+1
      N2=1
      KJ1 = N+1
      DO 40 L=1,KJ1
      MT=KJ1-L+1
   40 COF(MT)=XCOF(L)
C
C         SET INITIAL VALUES
C
   45 XO=.00500101
      YO=0.01000101
C
C         ZERO INITIAL VALUE COUNTER
C
      IN=0
   50 X=XO
C
C         INCREMENT INITIAL VALUES AND COUNTER
C
      XO=-10.0*YO
      YO=-10.0*X
C
C         SET X AND Y TO CURRENT VALUE
C
      X=XO
      Y=YO
      IN=IN+1
      GO TO 59
   55 IFIT=1
      XPR=X
      YPR=Y
C
C         EVALUATE POLYNOMIAL AND DERIVATIVES
C
   59 ICT=0
   60 UX=0.0
      UY=0.0
      V =0.0
      YT=0.0
      XT=1.0
      U=COF(N+1)
      IF(U) 65,130,65
   65 DO 70 I=1,N
      L =N-I+1
      TEMP=COF(L)
      XT2=X*XT-Y*YT
      YT2=X*YT+Y*XT
      U=U+TEMP*XT2
      V=V+TEMP*YT2
      FI=I
      UX=UX+FI*XT*TEMP
      UY=UY-FI*YT*TEMP
      XT=XT2
   70 YT=YT2
      SUMSQ=UX*UX+UY*UY
      IF(SUMSQ) 75,110,75
   75 DX=(V*UY-U*UX)/SUMSQ
      X=X+DX
      DY=-(U*UY+V*UX)/SUMSQ
      Y=Y+DY
      XSS=X
      YSS=Y
      IF(YSS.EQ.0.0D0)YSS=1.0D0
      IF(XSS.EQ.0.0D0)XSS=1.0D0
      ER1=DABS(DX/XSS)+DABS(DY/YSS)
      IF(ER1.GT.ER2)GO TO 78
      ER2=ER1
      XS=XSS
      YS=YSS
 78   IF(ER1-TOL)100,80,80
```

```
C
C        STEP ITERATION COUNTER
C
   80 ICT=ICT+1
      IF(ICT-500) 60,85,85
   85 IF(IFIT)100,90,100
   90 IF(IN-5) 50,95,95
C
C        SET ERROR CODE TO 3
C
   95 IER=3
      X=XS
      Y=YS
      ER1=ER2
  100 DO 105 L=1,NXX
      MT=KJ1-L+1
      TEMP=XCOF(MT)
      XCOF(MT)=COF(L)
  105 COF(L)=TEMP
      ITEMP=N
      N=NX
      NX=ITEMP
      IF(IFIT) 120,55,120
  110 IF(IFIT) 115,50,115
  115 X=XPR
      Y=YPR
  120 IFIT=0
122   IF(DABS(Y)-1.0D-8*DABS(X))135,125,125
  125 ALPHA=X+X
      SUMSQ=X*X+Y*Y
      N=N-2
      GO TO 140
  130 X=0.0
      NX=NX-1
      NXX=NXX-1
  135 Y=0.0
      SUMSQ=0.0
      ALPHA=X
      N=N-1
  140 COF(2)=COF(2)+ALPHA*COF(1)
  145 DO 150 L=2,N
  150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)
  155 ROOTI(N2)=Y
      ROOTR(N2)=X
      IF(ER1.GT.TOL)WRITE(6,554)N2,ER1
554   FORMAT(1X,'ERROR ON ',I3,'TH ROOT IS ',D10.3)
      ER2=1.0D+50
      N2=N2+1
      IF(SUMSQ) 160,165,160
  160 Y=-Y
      SUMSQ=0.0
      GO TO 155
  165 IF(N) 20,20,45
      END
      SUBROUTINE PLOTIT(X,NF,NPT,I1,I2,I3,MAXX,ISC,SCALE)
      DIMENSION X(MAXX,NF),IPL(97),IP(97),ICH(9),IS(3),XMX(9),XMN(9),F(9)
     1)
      REAL*8 XMAX,XMIN
      DATA ICH/1H*,1H+,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA IHI,IH,IHG,IHL/1HI,1H ,1H>,1H</
      DO201I=1,97
201   IP(I)=IH
      DO200I=1,8
      J=12*I+1
200   IP(J)=IHI
      IP(1)=IHI
      IF(NF-3)15,15,17
15    NP=NF
      GO TO 16
17    NP=3
16    DO111J=1,NF
      XMIN=X(1,J)
      XMAX=X(1,J)
      DO1I=2,NPT
      IF(XMAX-X(I,J))3,2,2
2     IF(XMIN-X(I,J))1,1,4
4     XMIN=X(I,J)
      GO TO 1
3     XMAX=X(I,J)
1     CONTINUE
      XMX(J)=XMAX
111   XMN(J)=XMIN
      NFF=NF
      IF(ISC.NE.1.OR.NF.EQ.1)GO TO 116
      XMAX=XMX(1)
      XMIN=XMN(1)
      DO117J=2,NF
      IF(XMAX.LT.XMX(J))XMAX=XMX(J)
      IF(XMIN.GT.XMN(J))XMIN=XMN(J)
117   CONTINUE
      DO118J=1,NF
      XMX(J)=XMAX
118   XMN(J)=XMIN
      NFF=1
116   IF(SCALE.EQ.1.0)GO TO 114
      DO115J=1,NFF
      XX=0.5*(XMX(J)*(1+SCALE)+XMN(J)*(1-SCALE))
      XMN(J)=0.5*(XMX(J)*(1.0-SCALE)+XMN(J)*(1.0+SCALE))
115   XMX(J)=XX
114   DO1113J=1,NFF
      XMAX=XMX(J)
      XMIN=XMN(J)
      XX=DABS(XMAX)
      XN=DABS(XMIN)
      IF(XX.LT.XN)GO TO 508
      MAX=1
      XSAV=XX
      GO TO 507
508   MAX=0
      XSAV=XN
507   IF(XMAX-XMIN-1.0D-8*XSAV)5,5,6
5     XMAX=XMAX+XX+1.0D00
      XMIN=XMIN-XN-1.0D00
      XX=DABS(XMAX)
      XN=DABS(XMIN)
      IF(XX.LT.XN)GO TO 518
      MAX=1
      XSAV=XX
      GO TO 6
518   MAX=0
```

```
       XSAV=XN
 6     XN=ALOG10(XSAV)
       JJ=XN
       XX=JJ
       JJ=JJ-1
       IF(XN-XX.NE.0.0.AND.XSAV.LT.1.0)JJ=JJ-1
       TEN=10.0**JJ
       K=XSAV/(TEN*8.0)
       IF(NMAX.EQ.1)GO TO 503
       IF(XMIN.LT.0.0)K=-(K+1)
       XX=XMAX/TEN
       KMIN=K*8
       XN=KMIN
 505   XN=XN+8.0
       IF(XN.LT.XX)GO TO 505
       KMAX=XN
       IF(KMAX*KMIN.LT.0)KMAX=-KMIN
       GO TO 1112
 503   IF(XMAX.GT.0.0)K=-(K+1)
       XX=XMIN/TEN
       KMAX=-K*8
       XN=KMAX
 504   XN=XN-8.0
       IF(XN.GT.XX)GO TO 504
       KMIN=XN
       IF(KMAX*KMIN.LT.0)KMIN=-KMAX
 1112  XMX(J)=KMAX*TEN
 1113  XMN(J)=KMIN*TEN
       IF(ISC.NE.1.OR.NF.EQ.1)GO TO 119
       DO120J=2,NF
       XMX(J)=XMX(1)
 120   XMN(J)=XMN(1)
 119   DO112J=1,NFF
       F(1)=XMN(J)
       XMIN=(XMX(J)-XMN(J))/8.0D00
       DO7I=1,8
 7     F(I+1)=F(I)+XMIN
       IF(F(1)*F(9).LT.0.0)F(5)=0.0
       WRITE(6,100)J,F
 100   FORMAT(5X,'FUNCTION',I2,7X,9E12.4)
 112   WRITE(6,106)ICH(J),IP,IP
 106   FORMAT(5X,'SYMBOL  ',1X,A1,18X,97A1,/,33X,97A1)
       DO8I=1,97
 8     IPL(I)=IP(I)
       DO113I=1,NF
 113   F(I)=96.0D00/(XMX(I)-XMN(I))
       DO600I=1,NF
 600   IS(I)=1
       IF(NP.EQ.1)ASSIGN 1001 TO NPPP
       IF(NP.EQ.2)ASSIGN 1002 TO NPPP
       IF(NP.EQ.3)ASSIGN 1003 TO NPPP
       DO9J=I1,I2,I3
       KMAX=1
       KMIN=97
       DO10I=1,NF
       K=F(I)*(X(J,I)-XMN(I))+1.5
       ISI=IS(I)
       IF(K)662,662,6621
 6621  IF(97-K)6331,633,633
 6331  K=97
       ICCC=1HG
       GO TO 644
 662   K=1
       ICCC=1HL
       GO TO 644
 633   ICCC=ICH(I)
```

```
644   IF(ISI-K)6011,601,601
6011  DO602L=ISI,K
602   IPL(L)=ICCC
      IF(KMIN.GT.ISI)KMIN=ISI
      IF(KMAX.LT.K)KMAX=K
      GO TO 603
601   DO604L=K,ISI
604   IPL(L)=ICCC
      IF(KMIN.GT.K)KMIN=K
      IF(KMAX.LT.ISI)KMAX=ISI
603   IS(I)=K
10    CONTINUE
      GO TO NPPP,(1001,1002,1003)
1001  WRITE(6,102)J,X(J,1),IPL
102   FORMAT(1X,I4,E12.4,16X,97A1)
      GO TO 18
1002  WRITE(6,103)J,X(J,1),X(J,2),IPL
103   FORMAT(1X,I4,2E12.4,4X,97A1)
      GO TO 18
1003  WRITE(6,104)J,X(J,1),X(J,2),X(J,3),IPL
104   FORMAT(1X,I4,3E9.3,1X,97A1)
18    DO19I=KMIN,KMAX
19    IPL(I)=IP(I)
9     CONTINUE
      WRITE(6,105)
105   FORMAT(1H1)
      RETURN
      END
```

```
      SUBROUTINE RESPON(X,V,N,GAMMA,XLAMDA,NP1)
      DIMENSION X(1),V(1),GAMMA(1),XLAMDA(1)
      REAL*8 XSAV,GAMMA,XLAMDA
      NM1=N-1
      NP1=N+1
      NPNP1=N+N+1
      NPNP2=N+N+2
      DO 19 I=1,NPNP1
19    XLAMDA(I)=0.0D00
      XSAV=0.0D00
      DO 20 K=1,NP1
      IF(N.EQ.1)GO TO 25
      DO 21 I=1,NM1
      J=NP1-I
21    XLAMDA(J)=XLAMDA(J-1)
25    CONTINUE
      DO 22 I=1,N
      J=NPNP2-I
22    XLAMDA(J)=XLAMDA(J-1)
      XLAMDA(1)=XSAV
      XLAMDA(NP1)=V(K)
      XSAV=0.0D00
      DO 23 I=1,NPNP1
23    XSAV=XSAV-GAMMA(I+1)*XLAMDA(I)
      IF(DABS(XSAV).GE.1.0D10)XSAV=0.0D00
20    X(K)=XSAV
      RETURN
      END
      SUBROUTINE ERROR(XREC,V,GAMMA,NP1,N,XLAMDA,XORG)
      DIMENSION XREC(1),V(1),XORG(1)
      DIMENSION  VVV(20)
      REAL*8 GAMMA(1),XLAMDA(1),AVGN,SUMV2,AVGQ
      CALL RESPON(XREC,V,N,GAMMA,XLAMDA,NP1)
      AVGN=0.0D00
      SUMV2=0.0D0
      DO26I=1,NP1
      SUMV2=SUMV2+XORG(I)*XORG(I)
      AVGQ=XORG(I)-XREC(I)
26    AVGN=AVGN+AVGQ*AVGQ
      AVGN=AVGN/SUMV2
      AVGQ=DSQRT(AVGN)
      AVGQ=100.0*AVGQ
      AVGN=100.0*AVGN
      WRITE(6,27)AVGN,AVGQ
27    FORMAT(1X,'PER CENT MEAN POWER ERROR OF RECONSTRUCTION',F8.3,///,
     11X,'PER CENT OF SQUARE ROOT OF POWER ERROR IN RECOSTRUCTION',F8.3)
      RETURN
      END



      SUBROUTINE BUILDA(A,Q,DEL,N,MAX)
      REAL*8 A(MAX,1),Q(1),DEL(1),PROD
      NP1=N+1
      NPNP2=N+N+2
      A(1,NP1)=1.0D00
      PROD=1.0D00
      DO312K=1,N
      I=NP1-K
      PROD=PROD/DEL(I)
      A(1,I)=PROD
312   A(K+1,NP1)=0.0D00
      DO313I=2,NP1
      DO313K=1,N
      J=NP1-K
313   A(I,J)=(A(I,J+1)-Q(J)*A(I-1,J+1))/DEL(J)
      DO314I=1,NP1
      DO314J=1,NP1
      A(I,J+NP1)=0.0D00
      A(I+NP1,J)=0.0D00
314   A(I+NP1,J+NP1)=A(I,J)
      WRITE(6,1005)
1005  FORMAT(1X,'A-MATRIX')
      CALL PRMAT(A,NPNP2,NPNP2,MAX)
      RETURN
      END



      SUBROUTINE PRMAT(A,N,M,NMAX)
      DOUBLE PRECISION A
C
C     THIS SUBROUTINE OUTPUTS DOUBLE PRECISION DOUBLE DIMENSIONED ARRAY
      DIMENSION A(NMAX,1)
      WRITE(6,1)
      DO2I=1,N
2     WRITE(6,3)(A(I,J),J=1,M)
3     FORMAT(1X,10D13.5)
      WRITE(6,1)
      WRITE(6,1)
1     FORMAT(/)
      RETURN
      END
      SUBROUTINE PRCVEC(A,N)
C
C     THIS SUBROUTINE PRINTS OUT A COMPLEX SINGLE DIMENSIONED ARRAY
C        A COMPLEX NUMBER OF THE FORM A + B J IS OUTPUTTED IN THE FORM
C                    ( A, B J)      WHERE J = SQUARE ROOT OF -1
      DIMENSION A(1)
      COMPLEX*16 A
      WRITE(6,2)
      WRITE(6,1)(A(I),I=1,N)
1     FORMAT(1X,1H(,D17.10,1H,,D17.10,3H J))
      WRITE(6,2)
      WRITE(6,2)
2     FORMAT(/)
      RETURN
      END
```

```
      SUBROUTINE PRVEC(A,N)
C
C     THIS SUBROUTINE OUTPUTS DOUBLE PRECISION SINGLE DIMENSIONED ARRAY
      DIMENSION A(1)
      DOUBLE PRECISION A
      WRITE(6,31)
      WRITE(6,1)(A(I),I=1,N)
1     FORMAT(1X,10D13.5)
      WRITE(6,31)
      WRITE(6,31)
31    FORMAT(/)
      RETURN
      END
```

```
      SUBROUTINE POLCON(C,R2,K,N)
C
C     A POLYNOMIAL CONSTRUCTION PROGRAM NEEDED FOR ZTOS
C
      DIMENSION C(1),R2(1)
      COMPLEX*16 C,R2,COMP
      REAL*8 DC(2)
      EQUIVALENCE (COMP,DC)
      NP1=N+1
      DO10I=2,NP1
10    R2(I)=0.0D00
      R2(1)=1.0D00
      DO4I=1,N
      COMP=C(I)
      IF(I.EQ.K.OR.(DC(1).EQ.0.0D0.AND.DC(2).EQ.0.0D0))GO TO 4
      DO2JJ=1,I
      J=I-JJ+1
2     R2(J+1)=R2(J+1)*C(I)+R2(J)
      R2(1)=R2(1)*C(I)
      CONTINUE
      RETURN
      END
```

APPENDIX B

HPMINV PROGRAM
LISTING

```
C           PROGRAM NAME: HPMINV
C              DOUBLE PRECISION MATRIX INVERSION PACKAGE
C
      DOUBLE PRECISION AA,A,P,Q,B,C,DA,G
      DOUBLE PRECISION THRES,PDIF,QDIF,TEM
      DOUBLE PRECISION FAC,FRAC,FFAC,ERROR
      DIMENSION AA(20,20),A(20,20),P(20,20),Q(20,20),B(20,20),
     1C(20,20),DA(20,20),G(20,20)
      COMMON /SOL/FRAC
      COMMON /PR/IPRINT
      COMMON /PROD/PDIF,QDIF
C
      MAX=20
      THRES=15.0D00
      WRITE(6,100)
      READ(5,240)N,ISP,ISQ,IPERT,ISLID,ICOND,ICORCT,IPRINT
      WRITE(6,140)N,ISP,ISQ,IPERT,ISLID,ICOND,ICORCT
      FRAC=1.0D00
      IF(IPERT.GT.N.AND.ISLID.EQ.1)FRAC=1.0D-04
      WRITE(6,190)
      DO 200 I=1,N
      READ(5,250)(AA(I,J),J=1,N)
      TEM=1.0D-09*AA(I,I)
      IF(ICOND.EQ.1)AA(I,I)=AA(I,I)+TEM
200   CONTINUE
      CALL MEQUAT(N,MAX,AA,A)
      IF(IPRINT.EQ.1)WRITE(6,350)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,A)
C                    SCALING METHOD
      PDIF=0.0D00
      QDIF=0.0D00
C              ROW SCALING
      IF(ISP.NE.1)GO TO 10
      IS=1
      CALL MSCALE(N,MAX,THRES,IS,PDIF,A,P)
      IF(IPRINT.EQ.1)WRITE(6,127)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,A)
      IF(IPRINT.EQ.1)WRITE(6,101)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,P)
      CALL DINV(N,MAX,P)
      IF(IPRINT.EQ.1)WRITE(6,102)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,P)
10    CONTINUE
C              COLUMN SCALING
      IF(ISQ.NE.1)GO TO 20
      IS=0
      CALL MSCALE(N,MAX,THRES,IS,QDIF,A,Q)
      IF(IPRINT.EQ.1)WRITE(6,103)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,Q)
      CALL DINV(N,MAX,Q)
      IF(IPRINT.EQ.1)WRITE(6,104)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,Q)
20    CONTINUE
C                   END OF SCALING
      IF(IPRINT.EQ.1)WRITE(6,105)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,A)
      CALL MEQUAT(N,MAX,A,C)
      IF(IPERT.EQ.0)GO TO 90
C
C     PERTURB MATRIX A, C=A+(FAC)*(DIAG A)=A+FAC*(DA)
C
      FAC=1.0D-04
      DO 80 INC=1,1
      FAC=FAC*1.0D-02
      WRITE(6,100)
      WRITE(6,500)FAC
      DO 85 I=1,N
      DO 85 J=1,N
      DA(I,J)=0.0D00
      B(I,J)=A(I,J)
      IF(IPERT.GT.N)GO TO 82
      IF(I.NE.IPERT)GO TO 83
82    CONTINUE
      IF(I.EQ.J)B(I,J)=A(I,J)+FAC*A(I,J)
      IF(I.EQ.J)DA(I,J)=A(I,J)
83    CONTINUE
      C(I,J)=B(I,J)
85    CONTINUE
      IF(IPRINT.EQ.1) WRITE(6,130)
      IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,C)
      IF(IPERT.GT.N.AND.ISLID.EQ.1)GO TO 90
      IF(IPERT.LE.N)GO TO 90
C
C     NOW 'C' IS THE PERTURBED MATRIX
C     INV A= INV C+ FAC*(INV C)*(DA)*(INV C)+(FAC)**2*((INV C)*
C     (DA))**2*(INV C)+...
C        NAPPRX=1  FIRST 2 TERMS OF THE SERIES FOR INV A ARE USED
C              =2  FIRST 3 TERMS OF THE SERIES FOR INV A ARE USED
C
      NAPPRX=2
      CALL DPERT2(N,MAX,NAPPRX,FAC,C,DA,B)
      WRITE(6,141)
      CALL PRMAT(N,MAX,B)
      GO TO 2295
C     APPLICATION OF PERTURBATION METHOD OVER------------------------------
90    CONTINUE
      CALL GKRPCT(N,MAX,C,B,DA)
      IF(PDIF-QDIF)666,666,667
666   CALL CORCT1(C,B,N,MAX,5,1)
      GO TO 668
667   CALL CORCT2(C,B,N,MAX,5,1)
668   CONTINUE
      IF(IPERT.EQ.0)GO TO 2295
      IF(IPERT.GT.N.AND.ISLID.EQ.1)GO TO 2291
      CALL DPERT1(A,B,N,MAX,FAC,IPERT)
2291  CONTINUE
      IF(IPERT.LE.N)GO TO 2295
      FFAC=FAC*1.0D-04
      DO 91 K=1,11
      WRITE(6,300)K
300   FORMAT(//,10X,'VALUE OF K =',I2,/)
      DO 89 I=1,N
      DO 89 J=1,N
89    G(I,J)=C(I,J)-FFAC*DA(I,J)
      IF(PDIF-QDIF)777,777,778
777   CALL CORCT1(G,B,N,MAX,5,0)
      GO TO 779
778   CALL CORCT2(G,B,N,MAX,5,0)
779   CONTINUE
      IF(K.LE.4)FFAC=FFAC*10.0D00
      IF(K.GT.4.AND.K.LE.7 )FFAC=FFAC*1.778279D00
      IF(K.GE.8)FFAC=FFAC*1.154782D0
```

```
91      CONTINUE
2295    CONTINUE
        IF(IPERT.EQ.0.AND.ICORCT.EQ.0)GO TO 783
        WRITE(6,580)
580     FORMAT(5X,'FINAL BID FOR IMPROVEMENT')
        FRAC=1.0D0
        NIT=N
        IF(PDIF-QDIF)781,781,782
781     CALL CORCT1(A,B,N,MAX,NIT,0)
        GO TO 783
782     CALL CORCT2(A,B,N,MAX,NIT,0)
783     CONTINUE
        WRITE(6,109)
        CALL PRMAT(N,MAX,B)
C
C             INVERSE AA = (INVERSE Q)*(INVERSE A)*(INVERSE P)
C
        CALL MEQUAT(N,MAX,B,C)
        IF(ISP.EQ.0.AND.ISQ.EQ.0)GO TO 92
        CALL MEQUAT(N,MAX,B,DA)
        IF(ISQ.EQ.1)CALL DMULT(N,N,N,MAX,FAC,0,Q,B,DA)
        IF(ISP.NE.1)CALL MEQUAT(N,MAX,DA,C)
        IF(ISP.EQ.1)CALL DMULT(N,N,N,MAX,FAC,0,DA,P,C)
92      CONTINUE
        IF(IPRINT.EQ.1)WRITE(6,107)
        IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,C)
        IF(PDIF-QDIF)93,93,94
93      CONTINUE
        CALL DMULT(N,N,N,MAX,FAC,0,A,B,G)
        IF(ISP.EQ.0)GO TO 95
        CALL DMULT(N,N,N,MAX,FAC,0,G,P,B)
        CALL DINV(N,MAX,P)
        CALL DMULT(N,N,N,MAX,FAC,0,P,B,G)
        GO TO 95
94      CONTINUE
        CALL DMULT(N,N,N,MAX,FAC,0,B,A,G)
        IF(ISQ.EQ.0)GO TO 95
        CALL DMULT(N,N,N,MAX,FAC,0,Q,G,B)
        CALL DINV(N,MAX,Q)
        CALL DMULT(N,N,N,MAX,FAC,0,B,Q,G)
95      CONTINUE
        IF(IPRINT.EQ.1)WRITE(6,108)
        IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,G)
        CALL MERROR(N,MAX,G,ERROR)
        WRITE(6,249)
        IF(IPERT.EQ.0)STOP
80      CONTINUE
100     FORMAT(////////////////////)
101     FORMAT(5X,'DIAGONAL SCALE MATRIX, P')
102     FORMAT(5X,'INVERSE P MATRIX')
103     FORMAT(5X,'DIAGONAL SCALE MATRIX, Q')
104     FORMAT(5X,'INVERSE Q MATRIX')
105     FORMAT(5X,'MATRIX A')
106     FORMAT(5X,'INVERSE A MATRIX')
107     FORMAT(5X,'INVERSE OF ORIGINAL MATRIX, AA')
108     FORMAT(5X,'PRODUCT OF ORIGINAL MATRIX, AA, AND ITS COMPUTED INVERS
       1E')
109     FORMAT(/,10X,'IMPROVED INVERSE MATRIX')
127     FORMAT(5X,'(ROW) SCALED MATRIX A')
130     FORMAT(5X,'C=A+ EPS*D  MATRIX')
141     FORMAT(5X,'INV(A) =  INV(C) + EPS*(   ) + EPS**2 *( )')
140     FORMAT(10X,'MATRIX DIMENSION =',I2,//,10X,'ISP =',I2,/,10X,'ISQ ='
       1,I2,/,10X,'IPERT =',I2,/,10X,'ISLID =',I2,/,10X,'ICOND =',I2,
       2/,10X,'ICORCT =',I2,//)
240     FORMAT(9I2)
249     FORMAT(5X,'**********************************************************
       1*****************************************')
250     FORMAT(3D25.13)
350     FORMAT(/////,5X,'ORIGINAL MATRIX, AA')
600     FORMAT(//,5X,'FAC=',D20.11,/)
        STOP
        END
```

```
      SUBROUTINE GKRDCT(N,MAX,X,Y,CCC)
      REAL*8 X(MAX,1),Y(MAX,1),A,B,C,D,E,DET,CCC(MAX,1)
      INTEGER NUM(2,20)
      COMMON /PR/IPRINT
      COMMON /PROD/PDIF,QDIF
      DO 6 I=1,N
      DO 6 J=1,N
6     Y(J,I)=X(J,I)
      A=1.0D0
      DO 43 I=1,N
      B=0.0D0
      L=I
      M=I
C
C        FIND LARGEST ENTRY A(L,M) IN LOWER DIAGONAL SUBMATRIX
C
      DO 18 J=I,N
      DO 18 K=I,N
      IF(DABS(Y(K,J)).LE.B)GO TO 18
      B=DABS(Y(K,J))
      L=K
      M=J
18    CONTINUE
C
C        INTERCHANGE ROWS
C
      IF(L.EQ.I)GO TO 24
      DO 23 J=1,N
      C=Y(L,J)
      Y(L,J)=Y(I,J)
23    Y(I,J)=C
C
C        INTERCHANGE COLUMNS
C
24    IF(M.EQ.I)GO TO 29
      DO 28 J=1,N
      C=Y(J,M)
      Y(J,M)=Y(J,I)
28    Y(J,I)=C
C
C        BEGIN SWEEP COLUMNS TO THE RIGHT
C        ARRAYS NUM(1,.) ,NUM(2,.) KEEP RECORD
C        OF ROW AND COLUMN INTERCHANGES
C
29    NUM(1,I)=L
      NUM(2,I)=M
      B=Y(I,I)
      Y(I,I)=A
      DO 42 J=1,N
      IF(J.EQ.I)GO TO 42
      C=-Y(I,J)
      Y(I,J)=0.0D0
      DO 41 K=1,N
      D=Y(K,I)*C
      E=Y(K,J)*B+D
C     IF(DABS(E).LT.1.0D-10*DABS(D))E=0.0D0
```

```
41     Y(K,J)=E/A
42     CONTINUE
43     A=B
C
C         RESTORE COLUMNS
C
       DO 58 I=2,N
       J=N+1-I
       K=NUM(2,J)
       IF(K.EQ.J)GO TO 52
       DO 51 L=1,N
       C=Y(K,L)
       Y(K,L)=Y(J,L)
51     Y(J,L)=C
52     K=NUM(1,J)
C
C         RESTORE ROWS
C
       IF(K.EQ.J)GO TO 58
       DO 57 L=1,N
       C=Y(L,K)
       Y(L,K)=Y(L,J)
57     Y(L,J)=C
58     CONTINUE
       DET=A
       WRITE(6,101)DET
101    FORMAT(/1X,'DET OF ORIG MATRIX IS ',D24.17)
       DET=1.0D00/DET
       DO 111 I=1,N
       DO 111 J=1,N
111    Y(I,J)=Y(I,J)*DET
       WRITE(6,102)
102    FORMAT(/1X,'INVFRSE MATRIX IS ')
C
C
C
       CALL PRMAT(N,MAX,Y)
C
C
       DO100I=1,N
       DO100J=1,N
       CCC(I,J)=0.0D00
       DO100K=1,N
       IF(PDIF-QDIF)66,66,67
66     CCC(I,J)=CCC(I,J)+X(I,K)*Y(K,J)
       GO TO 100
67     CCC(I,J)=CCC(I,J)+Y(I,K)*X(K,J)
100    CONTINUE
C
C
C
       WRITE(6,103)
       IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,CCC)
       CALL MERROR(N,MAX,CCC,ER)
103    FORMAT(/1X,'PRODUCT OF INVERSE AND ORIG MATRICES ')
       RETURN
       END
```

```
      SUBROUTINE CORCT1(X,Y,N,MAXX,NITER,IPR)
      DIMENSION X(MAXX,1),Y(MAXX,1),F(20,20),E(20,20),Z(20,20)
      COMMON /SOL/FRACI
      DOUBLE PRECISION X,Y,F,E,Z
      DOUBLE PRECISION FRAC,ER1,ER2,ER
      DOUBLE PRECISION FRACI,FRSAV
      WRITE(6,317)FRACI
      DO 290 ITER=1,NITER
      ER2=1.0D10
      ER=ER2
      FRAC=FRACI
      ER1=0.0D00
C
C     CALCULATE ERROR MATRIX,F=X*Y-I
C
      DO 210 I=1,N
      DO 210 J=1,N
      F(I,J)=0.0D00
      DO 200 K=1,N
200   F(',J)=F(I,J)+X(I,K)*Y(K,J)
      IF(I.EQ.J)F(I,J)=F(I,J)-1.0D00
      ER1=ER1+F(I,J)*F(I,J)
210   CONTINUE
      ER1=DSQRT(ER1/(N*N))
      WRITE(6,320)ER1
320   FORMAT(//,10X,'ORIGINAL RMSE = ',D15.8,/)
      IF(IPR.EQ.1)WRITE(6,300)
      IF(IPR.EQ.1)CALL PRMAT(N,12,F)
C
C     CALCULATE IMPROVED INVERSE,Y(IMPROVED)=Y-Y*F
C
      DO 230 I=1,N
      DO 230 J=1,N
      E(I,J)=0.0D0
      DO 220 K=1,N
220   E(I,J)=E(I,J)+Y(I,K)*F(K,J)
230   CONTINUE
      IF(IPR.EQ.1)WRITE(6,310)
      IF(IPR.EQ.1)CALL PRMAT(N,12,E)
235   CONTINUE
      ER2=ER
      CALL MEQUAT(N,20,Y,Z)
      DO 240 I=1,N
      DO 240 J=1,N
240   Z(I,J)=Y(I,J)-FRAC*E(I,J)
      CALL DMULT(N,N,N,20,FRAC,0,X,Z,F)
      CALL MERROR(N,20,F,ER)
      IF(ER.GE.ER2)GO TO 280
      FRSAV=FRAC
      FRAC=FRAC*0.5D00
      GO TO 235
280   CONTINUE
      DO 270 I=1,N
      DO 270 J=1,N
270   Y(I,J)=Y(I,J)-FRSAV*E(I,J)
290      CONTINUE
300   FORMAT(5X,'F MATRIX')
310   FORMAT(5X,'E MATRIX')
317   FORMAT(10X,'INITIAL FRAC= ',D11.4)
      RETURN
      END
      SUBROUTINE CORCT2(X,Y,N,MAXX,NITER,IPR)
      DIMENSION X(MAXX,1),Y(MAXX,1),F(20,20),E(20,20),Z(20,20)
      COMMON /SOL/FRACI
      DOUBLE PRECISION X,Y,F,E,Z
      DOUBLE PRECISION FRAC,ER1,ER2,ER
      DOUBLE PRECISION FRACI,FRSAV
      WRITE(6,317)FRACI
      DO 290 ITER=1,NITER
      ER2=1.0D10
      ER=ER2
      FRAC=FRACI
      ER1=0.0D00
C
C     CALCULATE ERROR MATRIX,F=Y*X-I
C
      DO 210 I=1,N
      DO 210 J=1,N
      F(I,J)=0.0D00
      DO 200 K=1,N
200   F(I,J)=F(I,J)+Y(I,K)*X(K,J)
      IF(I.EQ.J)F(I,J)=F(I,J)-1.0D00
      ER1=ER1+F(I,J)*F(I,J)
210   CONTINUE
      ER1=DSQRT(ER1/(N*N))
      WRITE(6,320)ER1
320   FORMAT(//,10X,'ORIGINAL RMSE = ',D15.8,/)
      IF(IPR.EQ.1)WRITE(6,300)
      IF(IPR.EQ.1)CALL PRMAT(N,12,F)
C
C     CALCULATE IMPROVED INVERSE,Y(IMPROVED)=Y-F*Y
C
      DO 230 I=1,N
      DO 230 J=1,N
      E(I,J)=0.0D0
      DO 220 K=1,N
220   E(I,J)=E(I,J)+F(I,K)*Y(K,J)
230   CONTINUE
      IF(IPR.EQ.1)WRITE(6,310)
      IF(IPR.EQ.1)CALL PRMAT(N,12,E)
235   CONTINUE
      ER2=ER
      CALL MEQUAT(N,20,Y,Z)
      DO 240 I=1,N
      DO 240 J=1,N
240   Z(I,J)=Y(I,J)-FRAC*E(I,J)
      CALL DMULT(N,N,N,20,FRAC,0,Z,X,F)
      CALL MERROR(N,20,F,ER)
      IF(ER.GE.ER2)GO TO 280
      FRSAV=FRAC
      FRAC=FRAC*0.5D00
      GO TO 235
280   CONTINUE
      DO 270 I=1,N
      DO 270 J=1,N
270   Y(I,J)=Y(I,J)-FRSAV*E(I,J)
290      CONTINUE
300   FORMAT(5X,'F MATRIX')
310   FORMAT(5X,'E MATRIX')
317   FORMAT(10X,'INITIAL FRAC= ',D11.4)
      RETURN
      END
      SUBROUTINE DPERT1(X,Y,N,MAXX,FAC,IPERT)
      DIMENSION X(MAXX,1),Y(MAXX,1),U(20),V(20),F(20,20)
      DOUBLE PRECISION X,Y,U,V,F
      DOUBLE PRECISION FAC,ALPHA,ER
      COMMON /PR/IPR
C
C
      IPR=2
      IPR=1
      WRITE(6,91)
      CALL DMULT(N,N,N,20,FAC ,0,X,Y,F)
      IF(IPR.GE.1)WRITE(6,95)
      IF(IPR.GE.1)CALL PRMAT(N,20,F)
      CALL MERROR(N,20,F,ER)
      WRITE(6,320)ER
320   FORMAT(//,10X,'ORIGINAL RMSE = ',D15.8,/)
C
C     CALCULATE IMPROVED INVERSE
C
      ER=FAC
      FAC=FAC*X(IPERT,IPERT)
      ALPHA=Y(IPERT,IPERT)
      ALPHA=1.0D0-FAC*ALPHA
      ALPHA=FAC/ALPHA
      IF(IPR.GE.1)WRITE(6,201)FAC,ALPHA
      FAC=ER
      DO 150 I=1,N
      U(I)=Y(I,IPERT)
      V(I)=Y(IPERT,I)
150   CONTINUE
      IF(IPR.EQ.2)WRITE(6,103)(U(I),I=1,N)
      IF(IPR.EQ.2)WRITE(6,105)
      IF(IPR.EQ.2)WRITE(6,103)(V(I),I=1,N)
      DO 240 I=1,N
      DO 240 J=1,N
240   F(I,J)=U(I)*ALPHA*V(J)
      IF(IPR.EQ.1)WRITE(6,300)
      IF(IPR.EQ.1)CALL PRMAT(N,20,F)
      DO 250 I=1,N
      DO 250 J=1,N
250   Y(I,J)=Y(I,J)+F(I,J)
      CALL DMULT(N,N,N,20,FAC ,0,X,Y,F)
      IF(IPR.GE.1)WRITE(6,97)
      IF(IPR.GE.1)CALL PRMAT(N,20,F)
      CALL MERROR(N,20,F,ER)
      WRITE(6,109)
      CALL PRMAT(N,MAXX,Y)
290      CONTINUE
      WRITE(6,91)
91    FORMAT(5X,'**********************************************************
     1***********************************')
95    FORMAT(5X,'APERT:  A*INV(C)',/)
97    FORMAT(5X,'APERT:  A*IMPROVED INV(C)',/)
103   FORMAT(2X,4(D25.18,3X))
105   FORMAT(//)
109   FORMAT(/,10X,'IMPROVED INVERSE MATRIX')
201   FORMAT(5X,'A *EPS=',D15.4,'ALPHA=',D15.4,/)
300   FORMAT(5X,'F MATRIX')
      RETURN
      END
      SUBROUTINE DPERT2(N,MAX,NAPPRX,FAC,C,DA,B)
      DOUBLE PRECISION C(MAX,1),DA(MAX,1),B(MAX,1),G(20,20)
      DOUBLE PRECISION FAC,PDIF,QDIF
      COMMON /PROD/PDIF,QDIF
      CALL GKRDCT(N,MAX,C,B,G)
      IF(PDIF-QDIF)51,51,52
51    CALL CORCT1(C,B,N,MAX,N,0)
      GO TO 53
52    CALL CORCT2(C,B,N,MAX,N,0)
53    CONTINUE
      CALL DMULT(N,N,N,MAX,FAC,1,B,DA,G)
      CALL DMULT(N,N,N,MAX,FAC,0,G,B,C)
      WRITE(6,118)
      CALL PRMAT(N,MAX,C)
      IF(NAPPRX.EQ.1)GO TO 87
      CALL DMULT(N,N,N,MAX,FAC,0,G,G,DA)
      CALL DMULT(N,N,N,MAX,FAC,0,DA,B,G)
      WRITE(6,121)
      CALL PRMAT(N,MAX,G)
87    CONTINUE
      DO 90 I=1,N
      DO 90 J=1,N
      IF(NAPPRX.EQ.1)B(I,J)=B(I,J)+C(I,J)
      IF(NAPPRX.EQ.2)B(I,J)=B(I,J)+C(I,J)+G(I,J)
90    CONTINUE
118   FORMAT(5X,'EPS* INV(C) *D*INV(C)')
121   FORMAT(5X,'EPS*INV(C) *D*INV(C)  + EPS**2*(INV(C)*D)**2 *INV(C)')
      RETURN
      END




      SUBROUTINE MTRANS(N,MAX,X)
      DOUBLE PRECISION X(MAX,1),Y(20,20)
C
      DO 1 I=1,N
      DO 1 J=1,N
1     Y(I,J)=X(J,I)
      DO 2 I=1,N
      DO 2 J=1,N
2     X(I,J)=Y(I,J)
      RETURN
      END




      SUBROUTINE MEQUAT(N,MAX,X,Y)
      DOUBLE PRECISION X(MAX,1),Y(MAX,1)
C
C             EQUATE MATRIX Y TO MATRIX X
C
      DO 1 I=1,N
      DO 1 J=1,N
1     Y(I,J)=X(I,J)
      RETURN
      END
      SUBROUTINE MSCALE(N,MAX,THRES,IS,DIF,X,Y)
      DOUBLE PRECISION X(MAX,1),Y(MAX,1),B(20,20)
      DOUBLE PRECISION THRES,ZERO,BTOT,THR,BTHR,YMAX,YMIN,DIF
C
C             ROW OR COLUMN SCALING OF MATRIX X
C                  IS=1, ROW SCALING
C                  IS=0, COLUMN SCALING
C
      ZERO=1.0D-50
      YMAX=-1.0D+10
      YMIN=1.0D+10
      IF(IS.EQ.0)CALL MTRANS(N,MAX,X)
      DO 30 I=1,N
      THR=-50.0
      DO 10 J=1,N
      IF(DABS(X(I,J)).GT.ZERO)GO TO 5
      B(I,J)=-50.0
      GO TO 10
5     CONTINUE
      B(I,J)-DLOG10(DABS(X(I,J)))
      IF(B(I,J).GT.THR)THR=B(I,J)
10    CONTINUE
      BTHR=THR-THRES
      BTOT=0.0D00
      ICT=0
      DO 20 J=1,N
      Y(I,J)=0.0D00
      IF(B(I,J).LT.BTHR)GO TO 20
      ICT=ICT+1
      BTOT=BTOT+B(I,J)
20    CONTINUE
      BTOT=BTOT/ICT
      Y(I,I)=10.**BTOT
      IF(BTOT.GT.YMAX)YMAX=BTOT
      IF(BTOT.LT.YMIN)YMIN=BTOT
      DO 30 J=1,N
      X(I,J)=X(I,J)/Y(I,I)
30    CONTINUE
      DIF=YMAX-YMIN
      IF(IS.EQ.0)CALL MTRANS(N,MAX,X)
      RETURN
      END
```

```
      SUBROUTINE MMULT(M,N,L,MAX,S,IS,A,B,C)
      DOUBLE PRECISION A(MAX,1),B(MAX,1),C(MAX,1),S,TEMP
C
C           DOUBLE PRECISION MATRIX MULTIPLICATION, C=A*B*S
C
      DO 10 I=1,M
      DO 10 J=1,L
      C(I,J)=0.0D00
      DO 10 K=1,N
      C(I,J)=C(I,J)+A(I,K)*B(K,J)
      IF(IS.EQ.1.AND.K.EQ.N)C(I,J)=S*C(I,J)
10    CONTINUE
      RETURN
      END
      SUBROUTINE PRMAT(N,MAX,A)
103   FORMAT(2X,4(D25.18,3X))
105   FORMAT(//)
      REAL*8 A(MAX,1)
      WRITE(6,105)
      M=((N-1)/4)+1
      DO 10 I=1,N
      WRITE(6,107)
      DO 15 L=1,M
      J1=(L-1)*4+1
      J2=J1+3
      IF(L.EQ.M)J2=N
      WRITE(6,103)(A(I,J),J=J1,J2)
15    CONTINUE
10    CONTINUE
      WRITE(6,104)
104   FORMAT(/////)
107   FORMAT(/)
      RETURN
      END
```

```
      SUBROUTINE MERROR(N,MAX,C,ERROR)
      DOUBLE PRECISION C(MAX,1),TEM,ATEM
      DOUBLE PRECISION ERROR,AER
      COMMON /ME/AER
C
C
      ERROR=0.0D00
      AER=0.0D0
      DO 1 I=1,N
      DO 1 J=1,N
      TEM=C(I,J)
      IF(I.EQ.J)TEM=C(I,J)-1.0D00
      ATEM=DABS(TEM)
      IF(ATEM.GT.AER)AER=ATEM
      ERROR=ERROR+TEM*TEM
1     CONTINUE
      ERROR=ERROR/(N*N)
      ERROR=DSQRT(ERROR)
      WRITE(6,100)ERROR,AER
100   FORMAT(12X,'(MAT-I):  RMSE=',D11.4,'   MAX=',D11.4)
      RETURN
      END
```

```
      SUBROUTINE DINV(N,MAX,X)
      DOUBLE PRECISION X(MAX,1)
C
C             INVERSION OF DIAGONAL MATRIX, X
C
      DO 1 I=1,N
      DO 1 J=1,N
1     IF(I.EQ.J)X(I,J)=1.0D00/X(I,J)
      RETURN
      END
```

- APPENDIX C -

In this appendix we present the details invclved in Example 2, page 10, of Chapter 2. The example deals with application of the identification technique to data obtaired from a single-stage transistor amplifier. The schematic and equivalent model of the circuit are repeated, for convenience, in Fig. C1.

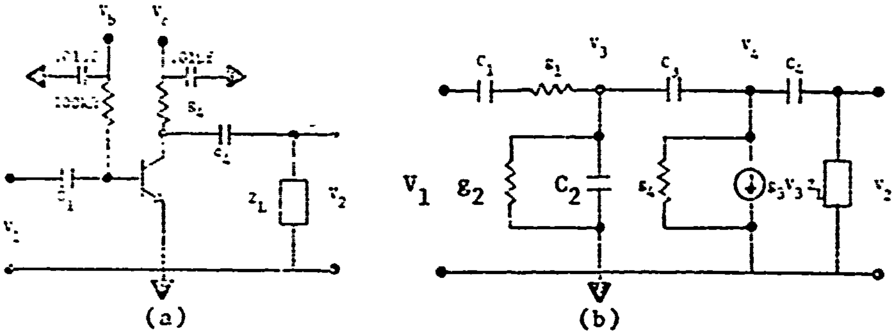


Fig. C1. (a) Schematic of Common-Emitter Amplifier Circuit.
(b) Equivalent Circuit Model

From Fig. C1b, the following equations may be written by inspection:

$$\begin{bmatrix} Y_1+g_2+(C_2+C_3)s & -C_3 s \\ g_3-C_3 s & Y_2+g_4+C_3 s \end{bmatrix} \begin{bmatrix} V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} Y_1 \\ 0 \end{bmatrix} V_1 \qquad \text{(C1)}$$

$$V_2 = \left(\frac{Z_L C_4 s}{Z_L C_4 s+1} \right) V_4 \qquad \text{(C2)}$$

where, by definition,

$$Y_1 \equiv \frac{g_1 C_1 s}{g_1+C_1 s} \qquad \text{(C3)}$$

$$Y_2 \equiv \frac{C_4 s/Z_L}{1/Z_L + C_4 s} \tag{C4}$$

[illegible]) and (C2) may be solved simultaneously, to yield the desired [illegible] r function, as follows. From (C1),

$$V_4 = \frac{\begin{vmatrix} Y_1+g_2+(C_2+C_3)s & Y_1 \\ g_3-C_3 s & 0 \end{vmatrix}}{\begin{vmatrix} Y_1+g_2+(C_2+C_3)s & -C_3 s \\ g_3-C_3 s & Y_2+g_4+C_3 s \end{vmatrix}} V_1 \tag{C5}$$

Substitution of equation (C5) into (C2) yields the required expression, i.e.,

$$H(s) = \frac{V_2}{V_1} = \left(\frac{Z_L C_4 s}{Z_L C_4 s+1} \right) \frac{\begin{vmatrix} Y_1+g_2+(C_2+C_3)s & Y_1 \\ g_3-C_3 s & 0 \end{vmatrix}}{\begin{vmatrix} Y_1+g_2+(C_2+C_3)s & -C_3 s \\ g_3-C_3 s & Y_2+g_4+C_3 s \end{vmatrix}} \tag{C6}$$

Simplification of the expression for H(s) requires unwieldy algebraic manipulation, and will therefore not be presented. However, it can be shown that for Z_L real H(s) assumes the following form:

$$H(s) = \frac{K s^2 (s+\beta_1)}{(s+\alpha_1)(s+\alpha_2)(s+\alpha_3)(s+\alpha_4)} \tag{C7}$$

For the particular transistor circuit shown in Fig. C1b, the following parameter values are assumed:

$C_1 = 0.01\mu f$ $C_3 = 5\ pf$ $g_1 = 4m℧$ $g_3 = 40m℧$ $Z_L = 1K\Omega$ (real)

$C_2 = 10.0pf$ $C_4 = 0.01\mu f$ $g_2 = 1m℧$ $g_2 = 0.5m℧$

Substituting these values in Equation (C6) and performing the required simplifications yields

$$H(s) = \frac{8(10^7)\ s^2(s-8000(10^6))}{(s+.033(10^6))(s+.080(10^6))(s+25.2(10^6))(s+1205.1(10^6))} \qquad (C8)$$

A Bode (corner) plot of Equation (C8) is given in Fig. C2. Inspection of this frequency response shows that the circuit is broad-band (i.e., it poles are separated by several decades). In order that the network transfer function be identified reliably, the spectrum of the input signal <u>must</u> contain sufficient energy concentrated in the vicinity of the network poles and zeros (for excitation and manifestation of these critical frequencies). However, it is not convenient to synthesize -- and realize in the laboratory -- an input having such characteristics.

Therefore, the network function may (and in this case will) be broken into constituent functions, each of which will be valid for a particular frequency range.

Inspection of Equation (C8) reveals that an adequate low-frequency description of H(s) is

$$H_L(s) = \frac{(21.04)\ s^2}{(s+.033(10^6))(s+.080(10^6))} \quad \text{(Low-freq.)} \qquad (C9)$$

which is valid up to 1 Mr/s. That is, Equation (C9) will closely approximate H(s) for radian frequencies below approximately 10^6 rad/sec. This observation can be seen clearly from the Bode plot in Fig. C2.

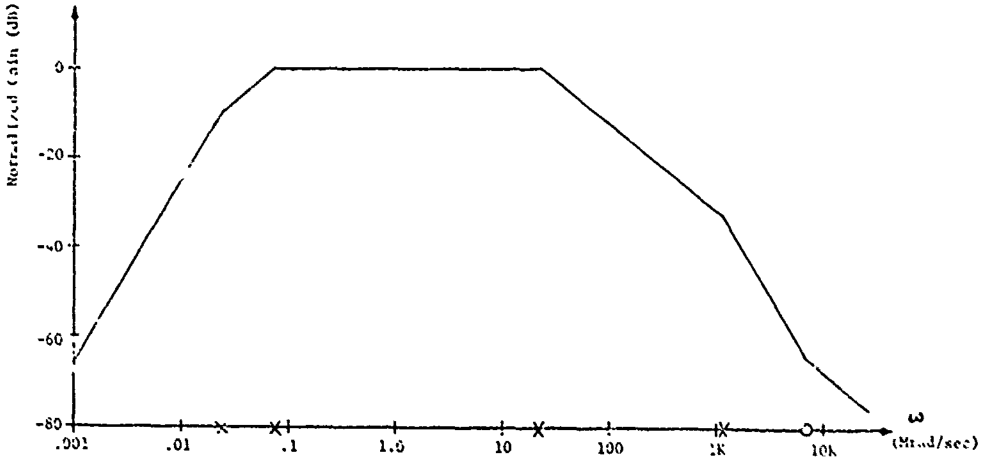
Through similar considerations, expressions describing the mid-high and high frequency characteristics of H(s) are obtained:

$$H_{MH}(s) = -\frac{531.1(10^6)}{(s+25.2(10^6))} \quad \text{(Mid to High Frequency Transistion)} \qquad (C10)$$

$$H_H(s) = \frac{8(10^7)\ (s-8000(10^6))}{(s+25.2(10^6))(s+1205.1(10^6))} \quad \text{(High Frequency)} \qquad (C11)$$

Equation (C10) will be valid in the frequency range from approximately 10^6 to 10^9 rad/sec, while Equation (C11) will be valid for frequencies from 10^9 rad/sec onward.

The identification technique may now be used to determine models for the network behavior by considering each of the three regions separately. Improvement in the methodology and reliability of identification of broad-band networks (systems) is being investigated under a new research task. For example, pre-filtering the output data in order to isolate the various frequency regions is now being pursued.



Midband gain = 26.2 dB

Fig. C2. Magnitude characteristic (Bode plot) of a wide band amplifier.

i) Low Frequency Region

Our approximate description of the low-frequency behavior of H(s) is given in eq. (C9). In order that a reliable model for this region be obtained, via the program IGRAM, several factors must be considered. First, a careful choice of the input must be made in order to excite the low frequency modes of the system. We need to isolate these modes of the response, and will therefore use an input signal whose spectral content is concentrated in the low frequency region. A satisfactory choice is a single triangular pulse of duration 125μsec. This signal will supply sufficient energy to the low frequency modes and relatively small amounts to the higher frequency modes.

Next, we must decide upon a sampling interval, Δ. A useful rule-of-thumb in making this choice is to sample at a frequency f_s at least ten times the highest frequency of interest. For the system under consideration, the highest frequency of interest is $0.013(10^6)$Hz*. Thus, a sampling interval $\Delta = 1/f_s =$ 0.25 μsec should be quite adequate. Notice that while we are sampling at an adequate rate for the low frequency modes, we are undersampling the high frequency modes. That is, the system as a whole is broad-band and we are sampling at a rate suitable only for the low frequency portion. Therefore, frequency aliasing can be expected to occur. The effect of this aliasing, however, (of the high frequency modes) appears as evenly distributed noise of relatively small power spectrum density.

An important, but less obvious, consideration is the total duration of the test record used in modeling. Whenever possible, a record long enough to have a few time constants, say one to four, of the slowest mode must be used. Using this criterion, a 1000 point record (MP1 = 1000) for the network under consideration should suffice.

ii) Mid to High Frequency Transistion

Our approximate description of the mid to high frequency transition behavior of H(s) is given by eq. (C10). Considerations similar to those made in the last section yield the following choices. Realizing that a narrowband signal must be

* It is unrealistic to expect that the design or test engineer know the exact frequencies of interest. However, it is assumed that he has some idea of the critical frequencies of the system.

used -- so as to excite only the mid-high frequency mode ($s=-25.2(10^6)$), an exponentially decaying sinusoid was chosen as the input signal. The center frequency of this input lies in the frequency range of interest. The sampling interval was chosen to be 0.01 μsec, and a 500-point record was used for modeling

In modeling this region, the option IBIAS = 1 was used. The reason for this choice is as follows. Due to the low-frequency modes, a transient response will appear in the system output in addition to the desired mid-frequency respons However, over the short duration of our record (5μsec) this slowly varying transient will appear relatively constant, resembling a d.c. bias. The option IBIAS = 1 allows the program to separate this "bias" and hence calculate a more reliable model for the mid-high transition range.

iii) High Frequency Region

The approximate high frequency description of H(s) is given in eq. (C11). The input signal used for network excitation must be narrowband (for previously mentioned reasons). Thus, a slowly decaying sinusoid with center frequency in the critical region was chosen. Five hundred points of input-output signals, with a sampling interval Δ = 0.00025μsec, were used for modeling.

Once the results for each of the frequency regions have been obtained, they may be used to synthesize the overall network response. This can be done by correctly combining the model descriptions of the various frequency regions. Details of such a synthesis will not be discussed.

APPENDIX D

s-Domain to z-Domain Conversion

Sampled Signal

When sampled at uniformly spaced time instants $k\Delta$, an analog signal $x(t)$ yields a numerical sequence $\xi = \{x_k\}$, where $x_k = x(k\Delta)$. To this numerical sequence we can associate a continuous-time signal $x^*(t) = \sum_{k=-\infty}^{\infty} x_k \delta(t-k\Delta)$, called the (ideally-)sampled signal. If the original signal is bandlimited by $1/2\Delta$ Hz, then $x(t)$ can be recovered from $x^*(t)$ through low-pass filtering, and the sampling process may be regarded as a one-to-one mapping. We define the Laplace transform of the sampled signal in the customary way; this gives

$$X^*(s) = \sum_{k=-\infty}^{\infty} x_k (e^{-s\Delta})^k \tag{D1}$$

Now, since the z transform of $\xi = \{x_k\}$ is

$$X(z) = \sum_{k=-\infty}^{\infty} x_k\, z^{-k} \tag{D2}$$

we make the extremely interesting observation that

$$X^*(s) = X(z)\Big|_{z=e^{s\Delta}} \tag{D3}$$

Note: It should be borne in mind that the substitution $z=e^{s\Delta}$ into $X(z)$ yields the Laplace transform of $x^*(t)$, not of $x(t)$. Under the condition of bandlimitedness (by $1/2\Delta$ Hz) this substitution yields a transform that agrees with $X(s)$ in a suitable neighborhood of $s=0$ in the s-plane.

We now focus attention on the matter of conversion of transfer functions from s-domain to z-domain and vice-versa. An exhaustive treatment is given in reference [17]. Here, we summarize three of the most widely used conversion techniques.

1. Logarithmic Pole-Zero Conversion

This technique uses the relation $z = e^{s\Delta}$, or $s = \frac{1}{\Delta}\,\mathrm{Ln}z$, upon the poles and zeros of the function under consideration. Thus

$$H(s) = \frac{s^{\ell} \prod_{i=1}^{m} (s+b_i)}{\prod_{i=1}^{n} (s+a_i)} \leftrightarrow H_1(z) = \Delta^{(n-\ell-m)} \frac{(z-1)^{\ell} \prod_{i=1}^{m} (z-\beta_i)}{\prod_{i=1}^{n} (z-\alpha_i)} \tag{D4}$$

where

$$\alpha_i = e^{-a_i\Delta} \qquad \text{D5a}$$

$$\beta_i = e^{-b_i\Delta} \qquad \text{D5b}$$

2. Pulse-Invariant Conversion

This technique has the merit that the response of $H(s)$ to an input $x_{pulse}(t) = \sum_{k=-\infty}^{\infty} x_k\, p(t-k\Delta)$, where $p(t)$ = square pulse over $(0,\Delta)$, coincides with the response of $H(z)$ to the sequence $\xi = \{x_k\}$. In many cases of practical interest $x_{pulse}(t)$ is an excellent approximation to $x(t)$; in such cases this technique of conversion promises close agreement of the response of $H(s)$ to $x(t)$ and of $H(z)$ to $\{x_k\}$, at the sampling instants. The conversion is described by

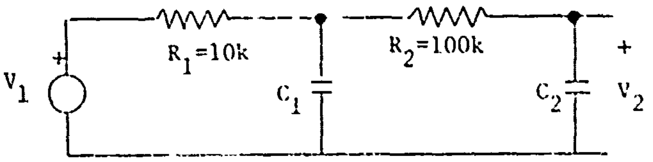
$$H(s) = \sum_{i=1}^{n} \frac{b_i}{s + a_i} \longleftrightarrow H_2(z) = \sum_{i=1}^{n} \frac{b_i\,(1 - e^{-a_i\Delta})}{a_i\,(1 - e^{-a_i\Delta} z^{-1})} \qquad \text{(D6)}$$

3. Impulse-Invariant Conversion

When this technique is used the response of $H(s)$ to $x^*(t)$ coincides with that of $H(z)$ to $= \{x_k\}$ (at the sampling instants). The conversion is described by

$$H(s) = \sum_{i=1}^{n} \frac{b_i}{s + a_i} \longleftrightarrow H_3(z) = \sum_{i=1}^{n} \frac{\Delta b_i}{1 - e^{-a_i\Delta} z^{-1}} \qquad \text{(D7)}$$

Example: Sampling Interval $\Delta = 5\mu s$



$C_1 = C_2 = 1000$pf

$$H(s) = \frac{1 \times 10^9}{s^2 + (1.2 \times 10^5)\, s + (1 \times 10^9)}$$

$$= \frac{1 \times 10^9}{(s+9009.8)(s+110{,}990.2)} \longleftrightarrow H_1(z) = \frac{0.025}{(z-0.95595)(z-0.57410)} \text{ by (D4)}$$

$$= \frac{9805.8}{s+9009.8} - \frac{9805.8}{s+110990.2} \leftrightarrow H_2(z)= \frac{0.047941z}{(z-0.95595)} - \frac{0.037628z}{(z-0.57410)} \quad \text{by (D6)}$$

$$\text{"} \qquad\qquad \text{"} \qquad \leftrightarrow H_3(z)= \frac{0.049029z}{(z-0.95595)} - \frac{0.049029z}{z-0.57410} \quad \text{by (D7)}$$

Remark: In 'IGRAM', conversion techniques 1 and 2 have been programmed. However, the present setting IZTS=1 (see page 44) leads to logarithmic conversion.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.